DTIC
ELECTE
FEB 2 3 1994
C

(1)

AFIT/GSM/LAS/93S-3

AD-A276 192

A COMPARISON OF THE *DISASTER*™ SCHEDULING SOFTWARE WITH A SIMULTANEOUS SCHEDULING ALGORITHM FOR MINIMIZING MAXIMUM TARDINESS IN JOB SHOPS

THESIS

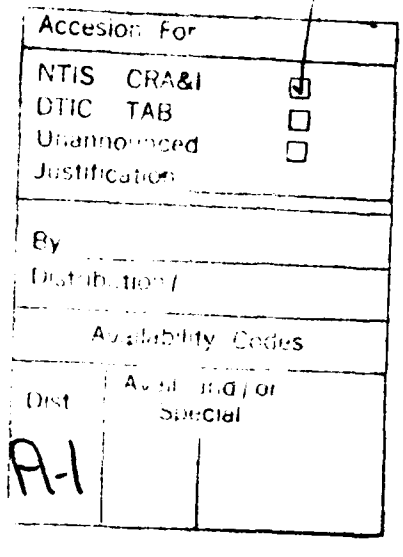
Barak J. Carlson, Captain, USAF
Christopher A. Lettiere, Captain, USAF
AFIT/GSM/LAS/93S-3

94-05640

Approved for public release; distribution unlimited

94 2 22 016

The views expressed in this thesis are those of the authors and do not reflect the official policy or position of the Department of Defense or the U.S. Government



AFIT/GSM/LAS/93S-3

A COMPARISON OF THE *DISASTER*™ SCHEDULING SOFTWARE WITH A SIMULTANEOUS SCHEDULING ALGORITHM FOR MINIMIZING MAXIMUM TARDINESS IN JOB SHOPS

Presented to the Faculty of the School of Logistics and Acquisition Management

of the Air Force Institute of Technology

Air University

In Partial Fulfillment of the

Requirements for the Degree of

Master of Science in Systems Management

Barak J. Carlson, B.S.
Captain, USAF

Christopher A. Lettiere, B.S.
Captain, USAF

September 1993

Approved for public release; distribution unlimited.

Table of Contents

page

List of Figures

page

List of Tables

Abstract

The Theory of Constraints (TOC) is the foundation for a computerized scheduling system called *DISASTER*™. Although this system has proven successful in many manufacturing settings, it has potential limitations due to the sequential heuristic process by which it schedules constraints. The objective of this thesis was to determine the extent to which these limitations impact the due date performance of schedules created by *DISASTER*™. This objective was addressed by developing an algorithm to simultaneously schedule multiple constraints in a job shop environment and provide the optimal schedule for minimized maximum tardiness. This algorithm was used to obtain solutions for a matrix of job shop scheduling problems, which were compared with solutions obtained by using *DISASTER*™. This comparison showed that *DISASTER*™ is capable of producing nearly optimal solutions for minimized maximum tardiness. but that this capability is highly dependent on proper constraint sequencing.

A COMPARISON OF THE *DISASTER*™ SCHEDULING SOFTWARE WITH A SIMULTANEOUS SCHEDULING ALGORITHM FOR MINIMIZING MAXIMUM TARDINESS IN JOB SHOPS

I. Introduction

General Issue

The U.S. Air Force (USAF) has found the Theory of Constraints (TOC) to be useful in many environments, ranging from common manufacturing challenges to battlefield tactics. In the operational arena, a study of USAF doctrine during Operation Desert Storm identified mobility as a constraint. This study recommended the use of TOC to provide procedural guidance and enhance the ability of F-16 units to move within striking distance of SCUD missile sights (Cummins, 1993:27). The Air Force has also successfully adopted TOC as a method for controlling aircraft modification programs at several Air Logistics Centers (ALCs). For example, at Ogden ALC the implementation of TOC concepts in an aircraft wheel repair facility reduced flow days by 75% while increasing throughput by 38% (Demmy and Petrini, 1992:6).

In an ALC, the complex flow of work is often modeled as a job shop. In a job shop, there may be any number of operations in a given job, and the workflow is multidirectional. As such, any given machine may be required to process the job more

than once in its production sequence (Baker, 1974:178). Job shop problems have been undertaken by many researchers over the past several decades with limited success.

> The general job-shop problem is a fascinating challenge. Although it is easy to state and to visualize what is required, it is extremely difficult to make any progress whatever toward a solution. Many proficient people have considered the problem, and all have come away empty-handed. Since this frustration is not reported in the literature, the problem continues to attract investigators, who just cannot believe that a problem so simply structured can be so difficult until they have tried it. (Conway, Maxwell, and Miller, 1967: 103)

This problem continues to draw attention because manufacturers view improvements in production scheduling as a necessity to future success in the global market (Wight, 1981:39). A 1981 study found that the typical manufacturing job in the U.S. spent about ten times more time waiting in queue than in actual production. Elimination of this unproductive time could improve inventory turn times by a factor of one hundred (Kanet, 1981:58). Clearly, yesterday's scheduling methods failed to maximize the effectiveness of production processes. However, since its introduction in the early 1980's, TOC has helped many job shop managers to achieve better system performance.

TOC simplifies the management of a system because it allows managers to control their processes by scheduling and monitoring only the system constraints. In a production process, the application of TOC usually results in the identification of no more than two or three constraints (Rose, 1993). A schedule for the constraints is much simpler than a schedule for the entire system, and may be obtained using mathematical solution techniques that are incapable of solving more computationally complex problems.

The usefulness of a schedule is often expressed in its ability to minimize or maximize a given performance measure. Common performance measures include schedule cost and schedule performance (Graves, 1981:648). As in many situations, cost limitations force a tradeoff in system performance. The objective of a solution technique must therefore be specified (i.e. minimize cost, maximize profit, minimize flow time, etc.) Due-date performance is often considered to be the most important schedule performance measure.

> Of the various measures of performance that have been considered in research on sequencing, certainly the measure that arouses the most interest in those who face practical problems of sequencing is the satisfaction of preassigned job due-dates. Equipment utilization, work-in-process inventory, and job flow-time are all interesting and more or less important, but the ability to fulfill delivery promises on time undoubtedly dominates these other considerations. (Conway, Maxwell, and Miller, 1967:229)

Prominent measures of due-date performance include: number of tardy jobs, total tardiness, mean tardiness, and maximum tardiness. Tardiness is defined as:

$$tardiness = max(completion\ time - due\ date,\ 0) \qquad (1)$$

Maximum tardiness has been chosen as the topic of this research, primarily because of its applicability to ALC operations. For example, if a number of jobs must be performed in order to complete the repair of an aircraft in the depot, it is the most late job that will determine when the repairs will be completed.

As part of a strategy to improve the performance of their production and modification schedules, Warner-Robins ALC and Ogden ALC have both entered into limited partnerships with the Avraham Y. Goldratt Institute, developer of a TOC based

computerized scheduling system called *DISASTER*™. At Warner Robins ALC alone, over $500,000 has been spent on software and training to implement *DISASTER*™ (Swartz, 1993).

Since its introduction in 1988, *DISASTER*™ has allowed many manufacturing companies to quickly create schedules that improve production throughput (Severs, 1991:166). However, the heuristic technique used to schedule constraints in a sequential manner is a common criticism of *DISASTER*™ because a user may generate different schedules for the same plant depending on the order in which the constraints are scheduled (Newbold, 1992:1). *DISASTER*™ creates a schedule for one constraint, and then builds schedules for subsequent constraints around the existing schedule. This sequential process introduces limitations which potentially sub-optimize the final schedule (Simons, 1992:2). The Goldratt Institute acknowledges the potential for eliminating this deficiency by simultaneously scheduling constraints, but has not yet pursued such an approach (Newbold, 1992:1). The USAF also recognizes the possible limitations of sequential scheduling, and the Production Policy Division of the Air Force Materiel Command Directorate of Logistics has sponsored this research effort to investigate the potential for improvement in this area.

Specific Problem Statement

The *DISASTER*™ solution algorithm is potentially limited by the sequential process which it uses to schedule constraints. The extent to which these limitations affect the maximum tardiness of the resulting schedules is unknown.

Research Objectives

The primary goal of this thesis is to determine the extent to which the limitations of sequential scheduling impact the due date performance of jobs scheduled by *DISASTER*™. Our research objectives are:

1. Search the literature to determine whether an optimal solution method exists for simultaneously scheduling multiple constraints in a job shop.
2. If an appropriate solution method cannot be found, develop a method to simultaneously schedule multiple constraints in a job shop.
3. Compare the maximum tardiness of schedules created by *DISASTER*™ with that of schedules created by a simultaneous scheduling method.
4. Determine the extent to which the performance of a simultaneous scheduling method is related to characteristics of the job shop.

Scope

We will address our research problem by identifying or developing an algorithm to simultaneously schedule multiple constraints in accordance with TOC. We will use this algorithm to obtain solutions to job shop scheduling problems, which we will compare with solutions for the same problems produced by *DISASTER*™.

Representative job shop scheduling problems must be mathematically modeled before a solution algorithm can be applied. A matrix of representative job shop scheduling problems for this research was developed by Captain Stewart W. James and Captain Bruno A. Mediate in thesis AFIT/GSM/LAS/93S-9. This matrix consists of problems containing two constraints in a variety of environments. The problems were developed through the manipulation of three primary design factors.

1. Plant type.
   a. A-plants, where one finished product is created from several raw materials.
   b. T-plants, where several finished products are created from several raw materials.
   c. V-plants, where several finished products are created from one raw material.

2. Percentage resource criticality factor (%RCF) of the least-loaded constraint (percent of maximum production capacity required).

3. The difference of percentage resource criticality factor (%ΔRCF) between the two constraints (difference in maximum production capacity required).

Details of how the design factors were combined to form problems are included in Appendix A and will be explained in more depth in Chapter 3.

Assumptions Several assumptions must be incorporated in our algorithm to account for the application of TOC and to make our results comparable to the schedules produced by *DISASTER*™. These assumptions involve the use of rods and buffers, both of which will be explained in more detail in the next chapter. The specific assumptions are:

1. All buffers are eight hours long.

2. A job is late when it breaches more than one-half of the shipping buffer.

3. All rods are four hours long (one half buffer length).

4. All constraints are known, and no other constraints will be identified through subordination.

5. All operations are available for processing any time after the ready time imposed by the rods between operations.

In addition to the assumptions required to make a valid comparison with *DISASTER*™, there are several additional assumptions that will apply to our model of

the job-shop. These assumptions are drawn from the works of Conway, Maxwell, and Miller (1967) and Baker (1974).

1. Only one of each machine type exists (i.e., no parallel processors).
2. All machines are continuously available and process material at a fixed rate.
3. Each operation has a constant processing time throughout the run of the model.
4. Setup times are either negligible or included in processing time (i.e., setups are not sequence dependent).
5. Operation pre-emption is not allowed.
6. Machines are not interchangeable and can process only one job at a time.

Approach

We will first review literature relevant to TOC and production scheduling, including an examination of the *DISASTER*™ scheduling program. We will then explore literature pertaining to scheduling problem classification, schedule generation, and mathematical optimization techniques. In the methodology, we will first outline the development of an algorithm to solve the scheduling problems. We will then review the methods of data collection and statistical analysis. The results of our research will then be presented, followed by conclusions and recommendations for further research.

II. Review of the Literature

The Theory of Constraints

It is well known that a chain is only as strong as its weakest link. TOC attempts to strengthen the chain (the manufacturing process) by focusing improvement efforts on the weakest links, which it designates as *constraints* (Simons and Moore, 1992:2).

Constraints. A constraint is anything that restricts the system from achieving a higher level of performance toward the organizational goal (Demmy and Petrini, 1992:7). Eliyahu Goldratt, founder of TOC suggests dealing with constraints through a process of five focusing steps:

1. **Identify the system's constraint(s).**
2. **Exploit the system's constraint(s).** In other words, determine how to achieve the maximum performance from the constrained resources.
3. **Subordinate all else to the above decision.** Processing by all non-constraint resources should be solely focused on supporting the constraint(s).
4. **Elevate the system's constraint(s).** Address the restrictions which cause the constraint(s), and attempt to eliminate them (i.e. increase capacity of constrained resource).
5. **If in the previous steps a constraint has been broken, go back to step 1, but do not allow inertia to cause a system's constraint** (Goldratt and Cox, 1992:303).

There are three basic types of constraints: external constraints, such as government regulations or market demand; internal policy constraints, resulting from organizational procedures and rules; and internal resource constraints, resulting when the demand for

a resource exceeds its physical limitations (Demmy and Petrini, 1992:8). The output of one constraint may later be the input for other constraints, a condition known as constraint interaction (Newbold, 1990:1).

Internal resource constraints can be classified as *bottlenecks* or *capacity constrained resources* (CCR) (Goldratt, 1990:187). A bottleneck is a resource that does not have adequate capacity to satisfy average work load demands (Goldratt, 1990:189). A CCR has adequate capacity to handle the average work load, but is overwhelmed by work load peaks due to a lack of protective capacity (Goldratt, 1990:187). Since it is only the lack of protective capacity that creates the capacity constraint for a CCR, it cannot be considered the prime constraint (Goldratt, 1990:189). Bottlenecks will be the only type of constraint dealt with in this research effort.

Drum-Buffer-Rope. The job shop scheduling technique used to implement Goldratt's five-step process, known as Drum-Buffer-Rope (DBR), is summarized here from Goldratt and Fox's 1986 book, *The Race*.

Once a constraint has been identified, we must exploit it and subordinate the rest of the system to it. This is where the ***drum*** comes into action. The constraint is scheduled for maximum efficiency, and this schedule (the drum) dictates the pace of the rest of the system. Essentially, the constraint is ***beating the drum*** for the rest of the resources to march to.

As the rest of the resources attempt to march to the constraint's drum they may encounter problems that temporarily halt overall production. These problems are dealt with through the use of three types of buffers: ***constraint buffers***, ***assembly buffers***,

and *shipping buffers*. If a resource feeding work to the constraint is disrupted, the constraint could potentially be starved for work, causing the entire system to lose throughput. A *constraint buffer* of material placed in front of the constraint will help prevent this occurrence. The size of the buffer is measured in terms of the amount of time it will take to process a given amount of material. For example, an "eight-hour" buffer would consist of the number of parts the constraint could process in an eight hour day. By providing this limited supply of work-in-process (WIP) before the constraint, we allow it to continue working through any disruption of preceding resources. Since the preceding resources have greater capacity than the constraint (or else they would also be constraints) they can use their excess capacity to replenish the buffer after the disruption has been corrected. *Assembly buffers* are placed ahead of assembly processes to hold non-constraint parts which are to be joined with parts from a constraint. Assembly buffers ensure that parts passed from the constraint continue to move through the production process without delay. Finally, a *shipping buffer* is placed at the end of the assembly process. This buffer absorbs disruptions from non-constraint processes occurring after processing by the final constraint which might result in failure to achieve customer due dates.

Finally, it is necessary to control the release of raw material into the plant at a pace that will not cause the growth of excess inventory beyond the limits of the buffers. This is done by establishing a control process, or *rope*, which only allows the release of raw materials into the plant at a pace that supports the constraint's drumbeat (Goldratt and Fox, 1986:96-100).

DISASTER™. The *DISASTER*™ scheduling software was developed by the Avraham Y. Goldratt Institute in 1986. *DISASTER*™ is a personal-computer based scheduling software package that implements the DBR scheduling technique. The user works interactively with the software to develop a schedule for the constraints in the plant. After the user describes the layout of his/her plant, *DISASTER*™ determines which processes represent constraints. If multiple constraints exist, *DISASTER*™ recommends a sequence in which to schedule the constraints. The choice of which constraint to schedule first often affects the quality of *DISASTER*™'s final schedule. The performance of *DISASTER*™ under these conditions is examined further in a thesis by Captains James and Mediate (AFIT/GSM/LAS/93S-9).

Ruins. Once a decision has been made on which constraint to schedule first, *DISASTER*™ constructs a diagram of the *ruins*, a Gantt-chart representation of the workload on the constraints under the assumption of infinite capacity (Goldratt, 1990:220). The ruins are then *leveled* (backward scheduled in reverse order based on due dates) to accommodate the limited capacity of the constraints (Goldratt, 1990:204). If the quantity of work exceeds current capacity, this levelling process will result in an infeasible schedule where work must begin in the past (Goldratt, 1990:205). To resolve this problem, the blocks are then forward scheduled from the present to represent the best effort that can be achieved by beginning this work immediately (Goldratt, 1990:205).

Buffers. The schedules created by *DISASTER*™ account for all three types of buffers previously described. *DISASTER*™ indicates a high probability that a job will

be late if the final constraint operation is completed less than one-half shipping buffer before the due date (Goldratt, 1990:206).

Rods. DISASTER™ uses the concept of *rods* to protect the flow of parts that must pass through multiple constraints, or the same constraint more than once (Goldratt, 1990:218). Rods prevent two constraint operations in the same job from being sequenced too close together. Scheduling constraint operations too close together would increase the probability that a system disruption could starve a constraint and delay job completions. Rods are heuristically defined to be one-half the length of the constraint buffer and their placement is determined by the processing time per part on the constraints (Goldratt, 1990: 218). As shown in Figure 1, the placement of the rods can allow an overlap of operations, or a gap between operations.

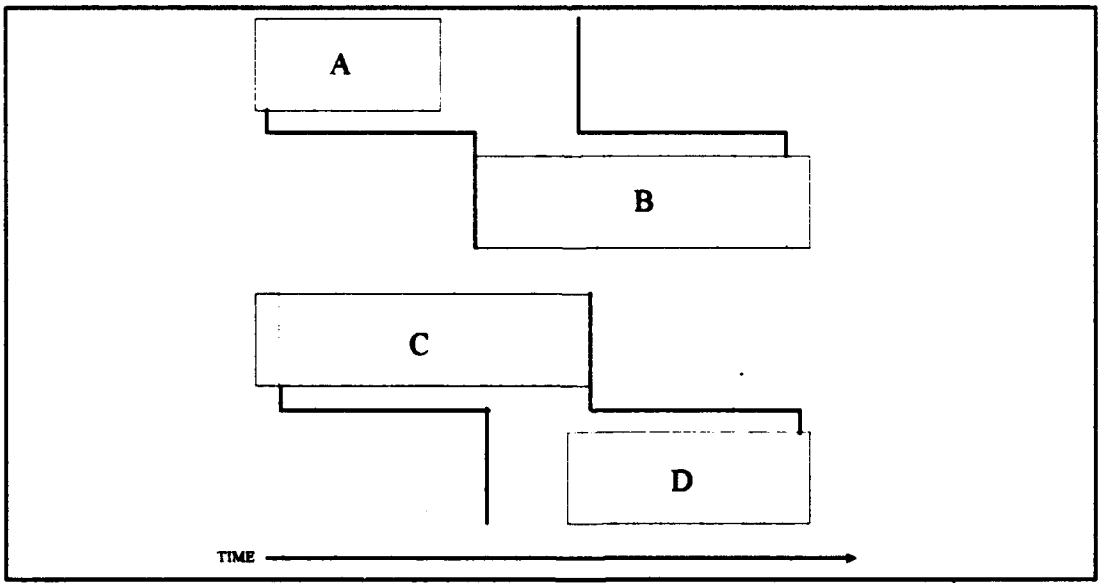


Figure 1: Placement of Rods

In the first example, two operations (A and B) each process ten parts. The processing time per part for operation B is greater than for operation A. A *forward rod* begins at the completion time of the first part in operation A and extends forward in time. A *backward rod* begins at the starting time of the last part in operation B and extends backward in time. Operations cannot be scheduled any closer together than is permitted by the rods. In this example, the placement of the rods creates a gap between the completion of operation A and the start of operation B.

The second example is analogous, except that the predecessor operation (C) has a longer processing time per part than the successor operation (D). In this particular case, the placement of the rods creates an overlap between the completion of operation C and the start of operation D. It must be noted that this overlap between operations can only occur when the operations are on different constraints and when the batch being processed consists of individual items which can be transferred to the next operation independently of the remaining items in the batch.

Consistent with TOC philosophy, *DISASTER*™ does not provide a schedule for non-constraints. Non-constraints work on a first-come-first-served basis according to the rate of material release (a function of the rope). *DISASTER*™ therefore produces executable schedules with significantly less input (or output) data than conventional scheduling packages (Severs, 1992:167-168).

Schedule Development

Theoretically, an infinite number of feasible schedules can be generated for any job shop problem, since an infinitely variable amount of idle time can exist between any two adjacent operations. However, when attempting to maximize any regular measure of scheduling performance, it is clearly desirable to eliminate any unnecessary idle time in the schedule (Baker, 1974: 181).

Semi-Active Schedules. Superfluous idle time can be eliminated by an adjustment called a *local-left-shift*: the movement of each operation to the earliest possible time without altering the operation sequences on any machine (Baker, 1974: 181). In this manner, a set of schedules with no superfluous idle time is created which is called the set of *semi-active schedules* (Baker, 1974:181). For any regular performance measure it is only necessary to consider the set of semi-active schedules, since the optimal schedule will be contained within this set (Conway, Maxwell, and Miller, 1967:109). This now leaves a finite set of schedules, since there is only one schedule for each of the finite number of possible operation sequences (Conway, Maxwell, and Miller, 1967:109). Although finite, this set is still too large to allow complete evaluation for most practical problems.

Active Schedules. A subset of the set of semi-active schedules can be created by an adjustment called a *global-left-shift* (Baker, 1974:183). A global-left-shift of an operation is any decrease in the starting time of the operation that does not increase the starting time of any other operation (Conway, Maxwell, and Miller, 1967:111). This allows an operation to shift past another operation into an interval of idle time, if

the interval is large enough to accommodate the shift (Conway, Maxwell, and Miller, 1967:111). This subset, known as the set of *active schedules*, dominates the set of semi-active schedules; which means that it is sufficient to consider only the set of active schedules when optimizing any regular measure of performance (Baker, 1974:185).

Schedule Generation. All schedule generation procedures operate on the set of operations to be scheduled, selecting operations one at a time, and assigning a starting time to each operation (Conway, Maxwell and Miller, 1967:112). The basic differences between schedule generation procedures are the order in which operations are selected, and the method in which the starting times are determined (Conway, Maxwell, and Miller, 1967:112). These differences will classify a schedule generation procedure as *single-pass* or *adjusting*. In a single-pass schedule generation procedure, the starting time of an operation is fixed when it is initially assigned, allowing full schedule generation with a single pass through the set of operations to be scheduled (Baker, 1974:187). In an adjusting schedule generation procedure, the initial assignment of an operation starting time is tentative and may be modified after subsequent operations have been scheduled (Baker, 1974:188).

Most computer programs use single-pass schedule generation procedures due to the difficulty of explicitly stating rules for adjustment (Conway, Maxwell, and Miller, 1967:112). This does not place any limits on the set of schedules that can be generated since there is a single-pass procedure capable of producing any schedule (Conway, Maxwell, and Miller, 1967:113).

Schedule generation procedures operate on the set of *candidate* operations: those operations which have no unscheduled predecessors (Baker, 1974:189). At each stage in the scheduling process, a separate set of scheduleable operations exists, along with a *partial schedule* consisting of all the operations that have already been assigned starting times (Baker, 1974:189). For each partial schedule, the potential start time of the next selected operation is determined by the completion time of the direct predecessor of that operation, and the earliest time the machine required by that operation is available (Baker, 1974:189).

Approaches for Solving Scheduling Problems

Attempts to solve complex scheduling problems fall into two general categories: heuristic solution procedures, and optimal solution procedures.

Heuristic Solution Procedures. Heuristic solution methods are used to create feasible schedules, but do not always provide the optimal schedule for a given performance measure. Heuristics are commonly applied to large problems where the enormous computational effort required to obtain an optimal solution makes other approaches impractical (Baker, 1974:195).

Common heuristic methods use a priority rule that is selected based on the management objective of the schedule. A schedule is built by sequencing the operations according to the selected priority rule. For the objective of minimizing tardiness, many priority rules such as earliest-due-date and minimum-slack-time have proven useful (Baker, 1984:195). The solution technique employed by *DISASTER*™

is heuristically based, and does not guarantee optimal schedule performance (Newbold, 1992:1).

Optimal Solution Procedures. Optimal solution procedures mathematically express the relationships between the operations in a problem and seek to generate a schedule satisfying all of these relationships while optimizing a measurable goal (Heizer, Render, and Stair, 1993:94).

Integer Programming. Integer programming is a general purpose approach used to optimally solve job shop scheduling problems (Baker, 1974: 192). This approach seeks to sequence individual operations to satisfy both sequencing requirements and noninterference restrictions (which address the inability of any machine to process more than one operation simultaneously) (Manne, 1960: 219). Technically, integer programming can be though of as including both pure integer programs and mixed integer programs. In pure integer programs, all decision variables must assume an integer value. In mixed integer programs, decision variables are both continuous and integer. The general job shop scheduling problem is usually formulated as a mixed integer program. A formulation of the problem identifies the objective to be optimized as well as the constraints that exist to characterize the problem.

In his 1960 paper "On the Job-Shop Scheduling Problem," Manne describes an integer programming formulation for the job shop problem. This formulation assumes each job requires each machine only once.

Let:

$x_{i,k}$ = the starting time of job i on machine k
$d_{i,k}$ = the processing time of job i on machine k

In order to satisfy the noninterference restrictions, the operations must precede each other by a sufficient quantity of time to allow the completion of one operation before another is scheduled. Accordingly, if there are two operations requiring machine k, then either

$$x_{1,k}-x_{2,k} \geq d_{2,k} \tag{2}$$

or

$$x_{2,k}-x_{1,k} \geq d_{1,k} \tag{3}$$

If job i precedes job j, then $x_{j,k}$ must be $d_{i,k}$ days after $x_{i,k}$. As such, the integer programming condition becomes

$$x_{i,k}+d_{i,k} \leq x_{j,k} \tag{4}$$

The sequence of operations is determined when all of the equations in the problem formulation are simultaneously solved for the values of $x_{i,k}$. For even small problems this set of constraints can quickly develop into an unmanageable system of inequalities. With only 4 machines and 10 jobs there are 220 variables and 390 constraint equations (Conway, Maxwell, and Miller, 1967:108). The increasing number of constraints and variables has forced many researchers to consider integer programming as impractical for most job shop problems. (Conway, Maxwell, and Miller, 1967:109).

Enumerative Methods. The enormous complexity of large integer programming problems has led to the development of specialized enumerative approaches to obtain optimal solutions (Patterson, 1984:855). These techniques enumerate efficiently,

requiring that only a small subset of the total number of possible solutions be examined individually (Levin and Kirkpatrick, 1975:349). This is accomplished by solving portions of the problem, temporarily ignoring certain restrictions or constraints. As the constraints are introduced back into the problem, a set of possible schedules is generated and ranked, revealing the best solution (Patterson, 1984:855).

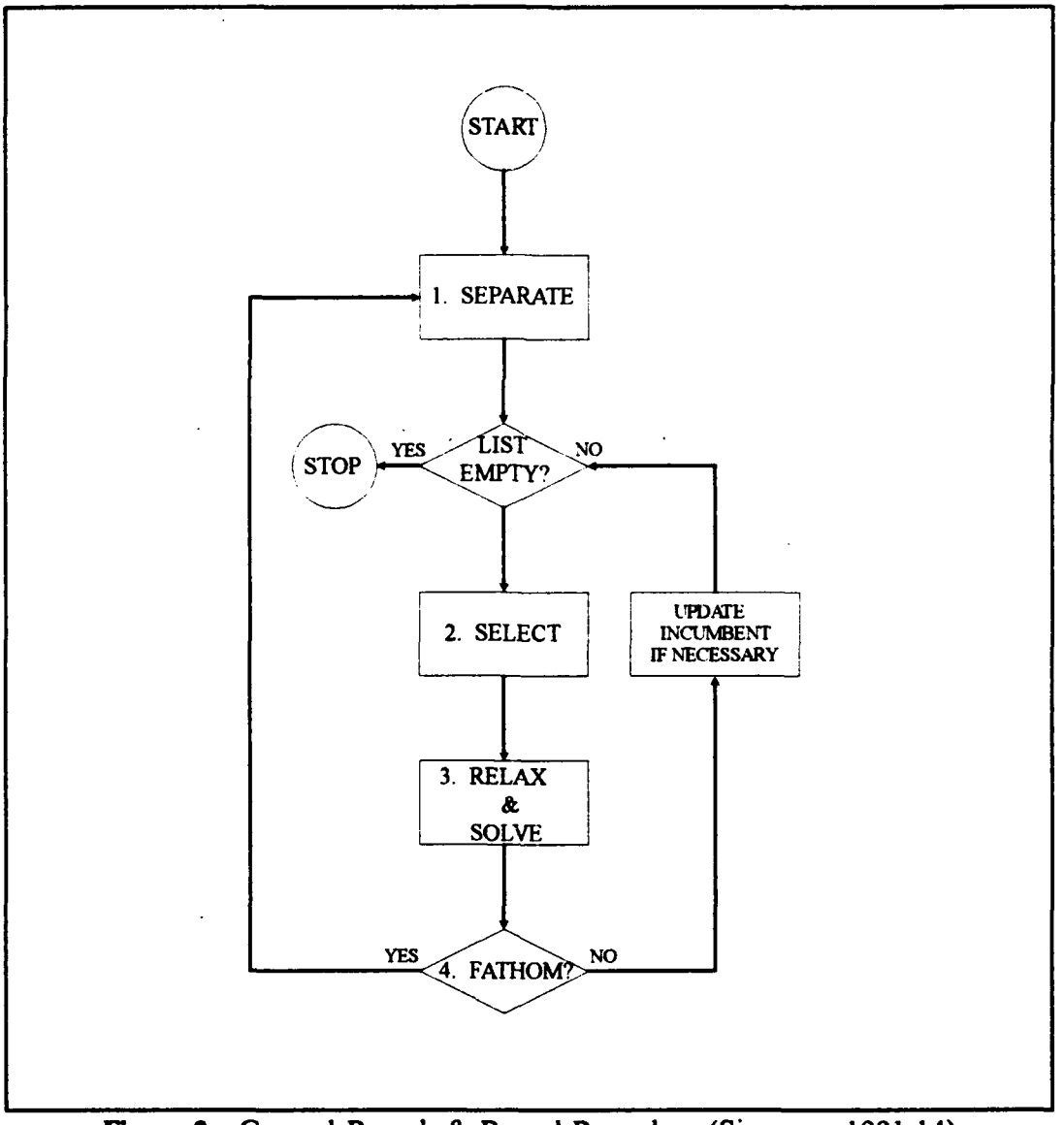


Figure 2: General Branch & Bound Procedure (Simpson, 1991:14)

Enumerative techniques are based on three fundamental concepts: separation, relaxation, and fathoming (Geoffrion and Marsten, 1972:465). These concepts are described below, and a strategy for their implementation is depicted in Figure 2.

Separation. Separation (Step 1) enables a "divide and conquer strategy" by dividing a problem into simplified subproblems that are easier to solve than the original problem. Subproblems must meet the following two criteria:

1. Every feasible solution of the parent problem is a feasible solution of exactly one of the subproblems.
2. A feasible solution of any of the subproblems is a feasible solution in the parent problem (Geoffrion and Marsten, 1972:466).

The most common way of separating a problem is by using contradictory constraints on a single variable (Geoffrion and Marsten, 1972:466). An example of this approach would be fixing one variable to a single value and solving the rest of the problem with this variable constant. The optimal solution for this subproblem can be selected (Step 2) as a candidate for the optimal solution to the parent problem while other subproblems are examined.

Relaxation. Relaxation (Step 3) determines a limit to the value of the objective function of a parent unrelaxed problem by loosening the constraints on the problem. This simplification of the parent problem implies the following relationships:

1. If the subproblem has no feasible solutions, the same is true for the parent problem.
2. If an optimal solution of the subproblem is feasible in the parent problem, then it is an optimal solution of the parent problem (Geoffrion and Marsten, 1972:467).

Relaxation can be performed by simply omitting one of the constraints or by eliminating the bounds or nonnegativity conditions on the variables in the parent problem (Geoffrion and Marsten, 1972:467).

Fathoming. Fathoming (Step 4) involves discarding subproblems from further consideration if:

1. The subproblem can be shown to have no feasible solution.
2. The subproblem has no feasible solution better than the current candidate solution to the parent problem (Geoffrion and Marsten, 1972:468).

If an optimal solution to the subproblem can be found, then the subproblem is completely accounted for, and there is no need for further separation (Geoffrion and Marsten, 1972:468).

By applying the techniques of separation, relaxation and fathoming to a problem, it is expected that the solution space (defined by the constraint set) can be systematically searched to find and verify the optimal solution to the problem (provided one exists). One enumerative technique in which these principles are applied is the *branch and bound* method, where solutions to valid subproblems provide a bound to the optimal solution of the initial problem.

Branch and Bound. Branch and bound (as applied to a scheduling problem) is an enumerative solution method which creates a "tree" of partial schedules. Each node in the tree represents an active schedule at a stage when more than one operation could be scheduled on a particular machine. Each "branch" emanating from a node represents the selection of one of the competing operations (Conway, Maxwell, and

Miller, 1967:117). Each branching operation can create as many new partial schedules as there are competing operations at that stage. Creation of all possible branches at each node will eventually result in a set of all possible feasible schedules (Stinson, Davis, and Khumawala, 1978:253). This set is too large to be practical for finding the optimal solution to even medium-sized problems. Therefore, partial schedules are fathomed using *lower bounds* and *upper bounds* as a means to reduce the time required to find and verify the optimal solution.

A *lower bound* based upon an existing partial schedule is calculated for every node, and represents the "best case" value for the objective function for all of the complete schedules which ultimately emanate from that node (Conway, Maxwell, and Miller, 1967:117). The value of a lower bound may not be feasible in a complete schedule, because it is often calculated through relaxation of the constraints. An *upper bound* is calculated after a complete schedule has been generated. It consists of the value of the objective function measure for the complete schedule (Conway, Maxwell and Miller, 1967: 117).

Techniques known as *search methods* use these bounds to converge on an optimal solution by evaluating the bounds of the branches and determining how the branch-and-bound tree is developed (Johnson, 1988:17). An example of a search method is the *last-in-first-out search* (LIFO). This method begins with the initial node in the solution tree, forms all branches descending from that node, and selects the branch with the lowest lower bound. A node is then formed from this branch, and the branching process is repeated. An upper bound (incumbent solution's objective

function value) is calculated when all operations have been sequenced. The LIFO search then backtracks along the node path to the nearest node that has a descending node that has not been extended. If this descending node's lower bound is superior to the upper bound, the branching process resumes (Johnson, 1988:24). Branches with lower bounds inferior to the upper bound are fathomed (i.e. eliminated from further consideration). Each time a new complete schedule is produced, its objective function value is compared to the current upper bound. If this new value is superior, the upper bound is adjusted and the schedule which produced it becomes the incumbent. If an optimal solution exists, this process will continue until an incumbent solution backtracks all the way to the initial node, and is thus identified as the optimal solution.

The value of fathoming lies in the fact that it reduces the number of alternative schedules which must be explicitly considered. A node whose lower bound exceeds the current upper bound cannot possibly produce an optimal solution. We know this because the existence of the upper bound indicates that we already have a schedule better than the best possible case which could be obtained from the node being fathomed.

Graph-Theoretical Method. One method of implementing the branch-and-bound solution technique for the job shop scheduling problem is the Graph-Theoretical machine sequencing algorithm outlined by Florian, Trepant, and McMahon in their 1971 paper "An Implicit Enumeration Algorithm for the Machine Sequencing Problem." Their technique involves a single-pass method which generates all active schedules, using a LIFO search method to obtain an optimal solution for regular

performance measures. The following sections summarize the portions of their formulation and solution technique relevant to this research effort.

A graphical notation is used to represent precedence conditions between operations. This notation creates a node for every operation, along with an initial node (0), and a terminal node (*). *Conjunctive arcs* are one-way arrows which connect nodes representing successive operations within each job. These arcs reflect precedence constraints between operations. Machine interference constraints that must be resolved to create a complete schedule are represented by opposing pairs of *disjunctive arcs*. Table 1 provides data for a sample problem consisting of two machines and three jobs.

Table 1. Sample Problem Data (Florian, Trepant, and McMahon, 1971:B783)

Operation	Job	Machine	Duration
1	1	1	4
2	1	2	3
3	2	2	2
4	2	1	3
5	3	1	1
6	3	2	2

Figure 3 depicts this problem in graph-theoretical form. Note that conjunctive arcs are shown between successive operations in the same job (such as operations 1 and 2). By contrast, disjunctive arcs join operations which must be processed on the same machine (such as operations 1 and 4), representing machine interference constraints. A solution for the problem is obtained by determining the optimal sequence of operations on each machine. This is sequence is created by "resolving" the disjunctive

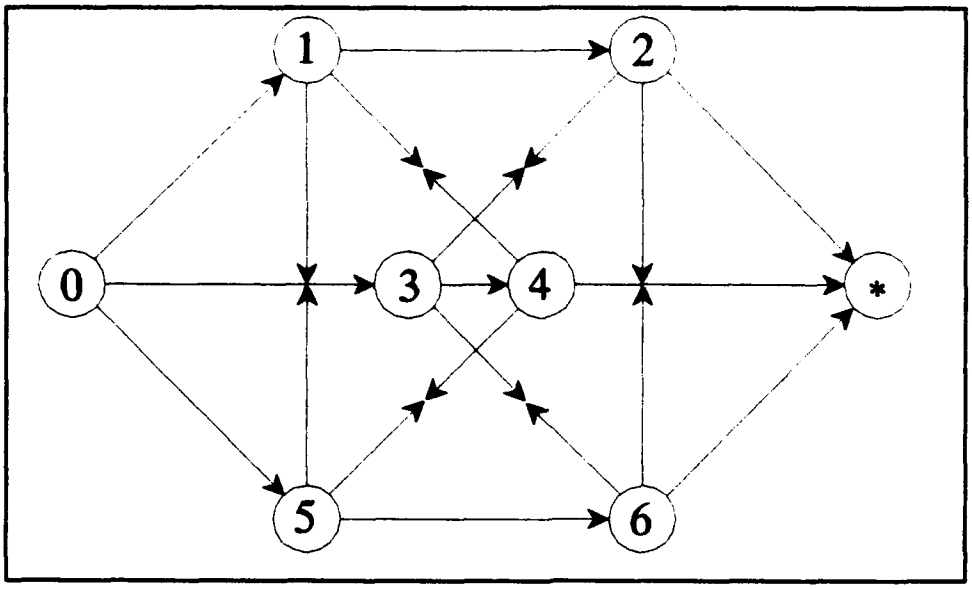


Figure 3: Graph-Theoretical Network (Florian, Trepant, and McMahon, 1971:B783)

arcs and replacing them with conjunctive arcs, indicating which of the two operations will be scheduled earlier.

Resolving the Disjunctive Arcs. A disjunctive arc is resolved when the sequence of the two operations connected by the arc is determined. An operation is *scheduled* on a machine when all disjunctive arcs connected to the operation have been resolved. To begin this process, it is necessary to identify the *initial cut* (C_0). A cut is defined as the set of all candidate operations (those with no unscheduled predecessors). In this example, C_0 consists of operations *1*, *3*, and 5, the only operations with no predecessors other than the initial node. A *generating set* is obtained from this cut, consisting of all operations in the cut on the same machine as the operation in the cut with the earliest possible finish time. A branch and bound tree node is created for each alternative within the generating set. Examination of Table 1

reveals that operation *5* has a duration of 1 time unit. Since all operations in the initial cut are able to start at time zero, operation *5* has the earliest possible finish time. The generating set consists of all operations in C_0 that are on the same machine as operation *5* (machine 1). Thus we see that the generating set for C_0 is $\{1,5\}$.

Since there is more than one operation in the generating set, it is necessary to decide which to sequence first on machine 1. A decision is made to schedule one of the nodes in the generating cut by calculating and comparing the lower bounds of the competing operations. The operation with the lowest bound is selected and scheduled by resolving all disjunctive arcs connected to the node. Because the lower-bound calculation method used by Florian, Trepant, and McMahon is not relevant to this research effort, it will not be reviewed here, and we will simply assume that operation *1* has the lowest bound.

Since operation *1* has the lowest bound, it becomes the first operation scheduled on machine 1. A new cut is now defined to begin selection of the next operation to schedule. This new cut (C_1) is created by removing operation *1* from the cut C_0, and adding its successor, operation *2* to the cut. A generating set is obtained from this cut as before, and the procedure repeats until all operations have been scheduled. This method is known as the *consecutive cut enumeration method.* As this method progresses, a tree of nodes begins to develop, such as in Figure 4. Florian, Trepant, and McMahon have proven that this method generates all active schedules for the job shop problem (Florian, Trepant, and McMahon, 1971:B-786). As mentioned

previously, it is sufficient to consider only the set of active schedules when seeking the optimal solution for any regular performance measure (Baker, 1974:185).

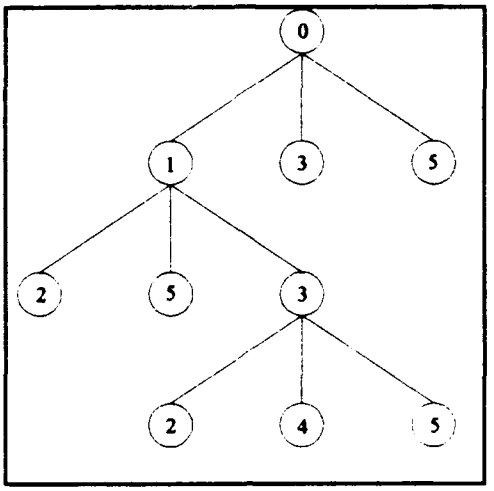


Figure 4: Tree of Nodes

A complete schedule is generated when all disjunctive arcs have been resolved. Figure 5 contains a network representation of a completed schedule that maintains the precedence and machine relationships for all operations in our example problem.

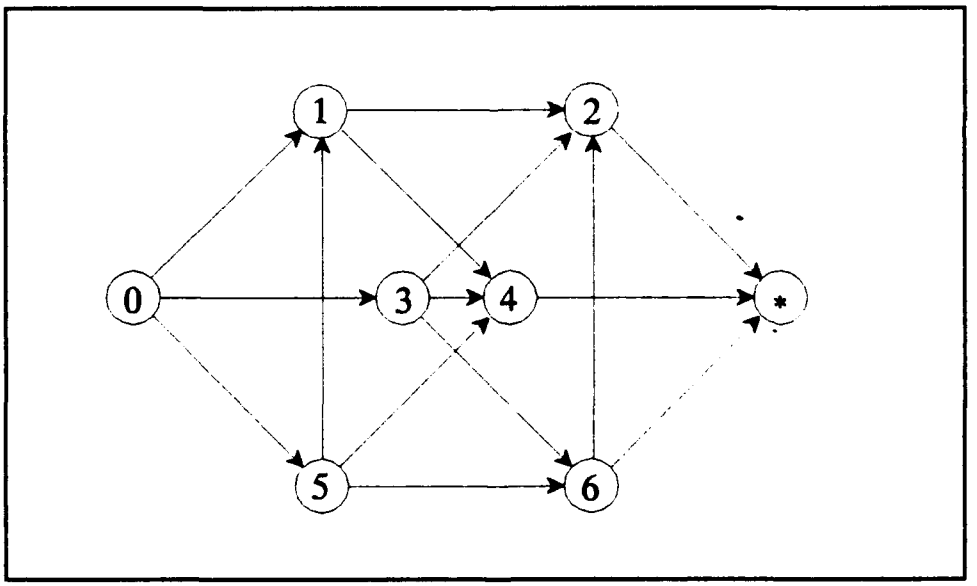


Figure 5: Resolved Graph-Theoretical Network

A Gantt chart corresponding to the schedule shown in Figure 5 is provided in Figure 6 to aid in the interpretation of the network.

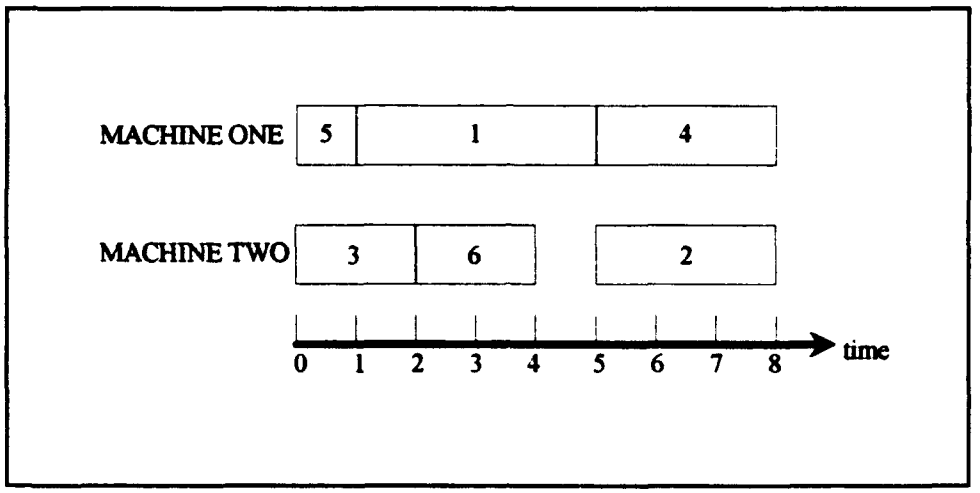


Figure 6: Gantt Chart for Resolved Network

Backtracking. Once a complete schedule has been generated, a value for the objective function is computed. (Florian, Trepant, and McMahon's objective function is to minimize makespan). This becomes the incumbent solution (upper bound). The optimality of this solution is then tested by backtracking through the tree of nodes. The lower bound of all nodes in the generating set of each cut are examined. If any of the lower bounds are less than the incumbent upper bound, then the corresponding operation is substituted for the operation originally scheduled from that cut. A new series of cuts is then defined, and the procedure repeats until we have backtracked all the way to the initial cut. If, however, a node's lower bound is greater than the incumbent upper bound, then the node is fathomed and the schedules which would have been produced from that node need never be explicitly considered. The sequence

producing the final upper bound is an optimal sequence for the desired management objective (Florian, Trepant, and McMahon, 1971:B-782 - B-790).

Summary

Our overview of TOC has reviewed the rationale allowing the scheduler to focus on the constraints in the system. Our review of the *DISASTER*™ software package demonstrated how this program applies TOC to generate a useable schedule. We also noted the potential shortcoming of *DISASTER*™ resulting from its heuristic solution method. We reviewed the classification of production scheduling problems, and the methods that exist to generate schedules for these problems. Finally, we examined methods for determining the solution to a production scheduling problem, with particular emphasis on the characteristics of the branch-and-bound solution technique and the Graph-Theoretical method for producing optimal solutions.

III. Methodology

In our survey of the existing literature, we did not find an optimal solution method that met our requirements. In accordance with our second research objective, we developed a solution method for simultaneously scheduling multiple constraints in a job shop. The first step in developing this method was to formulate the problem. Our formulation seeks to optimize due date performance by minimizing the maximum tardiness of all jobs. The algorithm we developed to solve this problem was an adaptation of the Graph-Theoretical method developed by Florian, Trepant, and McMahon for the general job shop. The basic node expansion logic of their method was retained, but new bounds were required due to the different objective function. Additional modifications were required to accommodate the differences between the general job shop and a job shop applying the principles of TOC.

Problem Formulation

Our problem formulation was adapted from Baker's 1974 integer programming formulation for the job shop problem. Our formulation uses the following notation.

Table 2: Function Notation

n	= the number of jobs
q_i	= the number of operations in job i
m	= the number of machines
i	= 1..n jobs
j	= 1..q_i operations
k	= 1..m machines
f_{ijk}	= completion time of operation j of job i on machine k
h_i	= due date of job i
d_{pqk}	= processing time of operation q of job p on machine k
g_{ijk}	= the ready time of operation j of job i on machine k
X	= very large positive number
C	= the set of all jobs
$y_{(ip)(jq)k}$	= 1 if operation j of job i precedes operation q of job p on machine k, else = 0

Objective Function:

$$min\left(\max_{i \in C}\left(\max((f_{iq,k}-h_i), 0)\right)\right) \quad (5)$$

Subject To:

$$f_{ijk}-d_{ijk} \geq g_{ijk} \quad \forall\, i,j,k \quad (6)$$

$$f_{pqk}-f_{ijk}+X(1-y_{(ip)(jq)k}) \geq d_{pqk} \quad \forall\, i,j,p,q,k \quad (7)$$

$$f_{ijk}-f_{pqk}+X(y_{(ip)(jq)k}) \geq d_{ijk} \quad \forall\, i,j,p,q,k \quad (8)$$

$$f_{ijk}-d_{ijk} \geq 0 \quad \forall\, i,j,k \quad (9)$$

The objective is to minimize the maximum tardiness (the difference between due date and completion) across the set of all jobs. Equation (7) states that no operation

can start any earlier than its ready time. Equations (8) and (9) prevent the scheduling of two operations on the same machine simultaneously. Equation (10) prevents scheduling of operations in the past. The g_{ijk} are used to accommodate the rods.

Algorithm Development

Our algorithm was based on the Graph-Theoretical machine sequencing algorithm developed by Florian, Trepant, and McMahon described in the previous chapter. This method was selected because their solution technique was easily adaptable to the objective of minimizing maximum tardiness by simply using a different lower bound. The bound we developed for this research effort is calculated by assigning each operation the due date of its job and sequencing the operations by earliest due date (EDD) with ties broken arbitrarily. The value of this lower bound reflects the minimum possible maximum tardiness at each point in sequence of operations. A proof of the validity of this lower bound is contained in the following sections.

Assumptions: All operations are ready for processing at t=0 (referred to later as the *assumption of readiness*).

Notation:

C = the set of all operations
i,j = two operations within C
B_i = the set of operations preceding operation i in the current schedule
f_i = the completion time of operation i
h_i = the due date of the job containing operation i
T_i = the tardiness of operation i
d_i = the processing time of operation i

Single Machine Tardiness Determination Hypothesis: On a single machine, for all jobs, tardiness of a job will be determined by the maximum tardiness of the operations in that job.

Single Machine Tardiness Determination Proof: Each job consists of multiple operations and is assigned a job due date. Assume that there exists a job, A, consisting of two operations, i and j. Job A will be complete upon the completion of operations i and j. It can therefore be stated that the due date for the completion of operations i and j is the due date for the completion of job A. Tardiness is defined as the positive difference between the completion of an operation or job and its due date. Since the due date is the same for both operations i and j, the tardiness of each operation will be determined by the operation's finish time. In the two-operation example with job A, there are only four possible outcomes:

1. If $f_i \leq h_i$ and $f_j \leq h_i$ then $T_A = 0$
2. If $f_i > f_j$ and $f_i > h_i$, then $T_A = T_i$
3. If $f_j > f_i$ and $f_j > h_j$, then $T_A = T_j$
4. If $f_i = f_j$ and $f_i > h_i$ then $T_A = T_i = T_j$

The above example can be extended to jobs consisting of more than two operations; however, in all cases the tardiness of a job is determined by the tardiness of the most tardy operation.

Single Machine Tardiness Minimization Hypothesis: Since the tardiness of a job will be determined by the tardiness of its most tardy operation, the minimized

maximum tardiness of multiple jobs can be determined by minimizing maximum tardiness of the operations of these jobs.

Single Machine Tardiness Minimization Proof: The following proof of minimizing maximum tardiness with EDD sequencing is adapted from Baker (1974).

An initial sequence, S, may be formed by scheduling the operations in any order. The assumption of readiness results in a schedule with no idle time - a continuous period of activity on the machine from t = 0 until the completion of the last scheduled operation. The makespan of this activity on the machine will remain the same regardless of the sequence of operations.

If the operations are not in EDD sequence, there will exist two operations, i and j, with j directly following i such that $h_i > h_j$. A new sequence, S', can be constructed in which i and j are interchanged and all other operations complete in the same time as in sequence S.

The tardiness of the two operations in sequence S will be determined as the positive difference between the completion of an operation and its due date:

$$T_i(S) = max(f_{Bi}+d_i-h_i, 0) \qquad (10)$$

$$T_j(S) = max(f_{Bi}+d_i+d_j-h_j, 0) \qquad (11)$$

The tardiness of the two operations in sequence S' will be similarly determined as:

$$T_i(S') = max(f_{BI}+d_j+d_i-h_i, 0) \quad (12)$$

$$T_j(S') = max(f_{BI}+d_j-h_j, 0) \quad (13)$$

If neither operation has tardiness > 0, there is no benefit or penalty in terms of minimized maximum tardiness by interchanging the two operations. If both operations have positive tardiness, the above relationships show that:

$$T_j(S) > T_i(S') \quad (14)$$

and

$$T_j(S) > T_j(S') \quad (15)$$

The tardiness of operation j in sequence S is greater than the tardiness of i or j in sequence S'. If operation i or operation j are not the most tardy operations in the sequence S, the value for T_{max} will be unchanged in sequence S'. This procedure may be repeated for all operations in the sequence where the succeeding operation's due date is earlier than that of the preceding operation. At each iteration, the value of T_{max} will remain the same or be reduced. After all of these cases have been resolved, the resulting sequence will minimize maximum tardiness for all operations in the schedule.

Multiple Machine Hypothesis: The procedure that we have just reviewed for sequencing operations on a single machine can be used as a bound in the multiple machine case by ignoring operational precedence constraints and scheduling each machine independently. By scheduling the remaining operations of all jobs on each

applicable machine by EDD, we will minimize the maximum tardiness of each job relative to each machine. The value for the bound will then be the maximum tardiness of all jobs among all machines.

Multiple Machine Proof: As presented in Geoffrion and Marsten (1972), any constrained optimization problem, *P*, can be relaxed by loosening its constraints, resulting in a new problem, *P'*. In the case of our bound, the problem of sequencing the operations among both machines, P, is simplified by relaxing the operational precedence constraints and the machine interference constraints. The new problem, P', requires that we minimize the maximum tardiness among each machine independently when all operations are available for processing at T=0. The property of relaxation implies that the minimal value of our performance measure for *P* is no less than the minimal value of the performance measure for *P'*. Additionally, if an optimal solution of *P'* is feasible in *P*, then it is an optimal solution for P.

Statement of the Basic Algorithm

Our algorithm, as adapted from the algorithm outlined in Florian, Trepant, and McMahon's 1971 paper to reflect the objective of minimizing maximum tardiness, is presented below.

1. Problem initialization
    a. Designate the initial cut, a set consisting of all operations whose only predecessor is the initial node.
    b. Compute earliest start times for all operations.

c. Set upper bound = ∞.

2. Extract Generating Set from Cut

   a. Determine the operation in the cut with the earliest finish time.

   b. All operations in the cut on the same machine as the above operation are in the generating set.

   c. If there are no operations in the generating set, then go to step 8.

   d. If there is only one operation in the generating set, then go to step 6.

   e. If there is more than one operation in the generating set, then continue.

3. Compute the lower bound for every operation in the generating set.

   a. Assume the selected operation is scheduled.

   b. Sequence all unscheduled operations by earliest due date.

   c. Calculate the tardiness for each job.

   d. Record maximum tardiness. This is the lower bound for the selected operation.

4. Select the operation in the generating set with the lowest value for the lower bound. If this lower bound $\geq$ the incumbent upper bound, then go to step 7; otherwise schedule the selected operation next on its designated machine.

5. Set the lower bound for the selected operation = ∞.

6. Remove the selected operation from the cut, and add any successors to the selected operation to the cut. Compute new earliest start and finish times for each unscheduled operation. Go to step 2.

7. If the current cut is the initial cut, then go to step 9; otherwise backtrack to the previous cut and go to step 4.

8. A complete sequence has been generated. Compare the maximum tardiness of this sequence to the incumbent upper bound. If this sequence results in a lower maximum tardiness, then the maximum tardiness resulting from this sequence is the new upper bound. Go to step 7.

9. The last complete sequence generated was the optimal sequence for minimizing maximum tardiness.

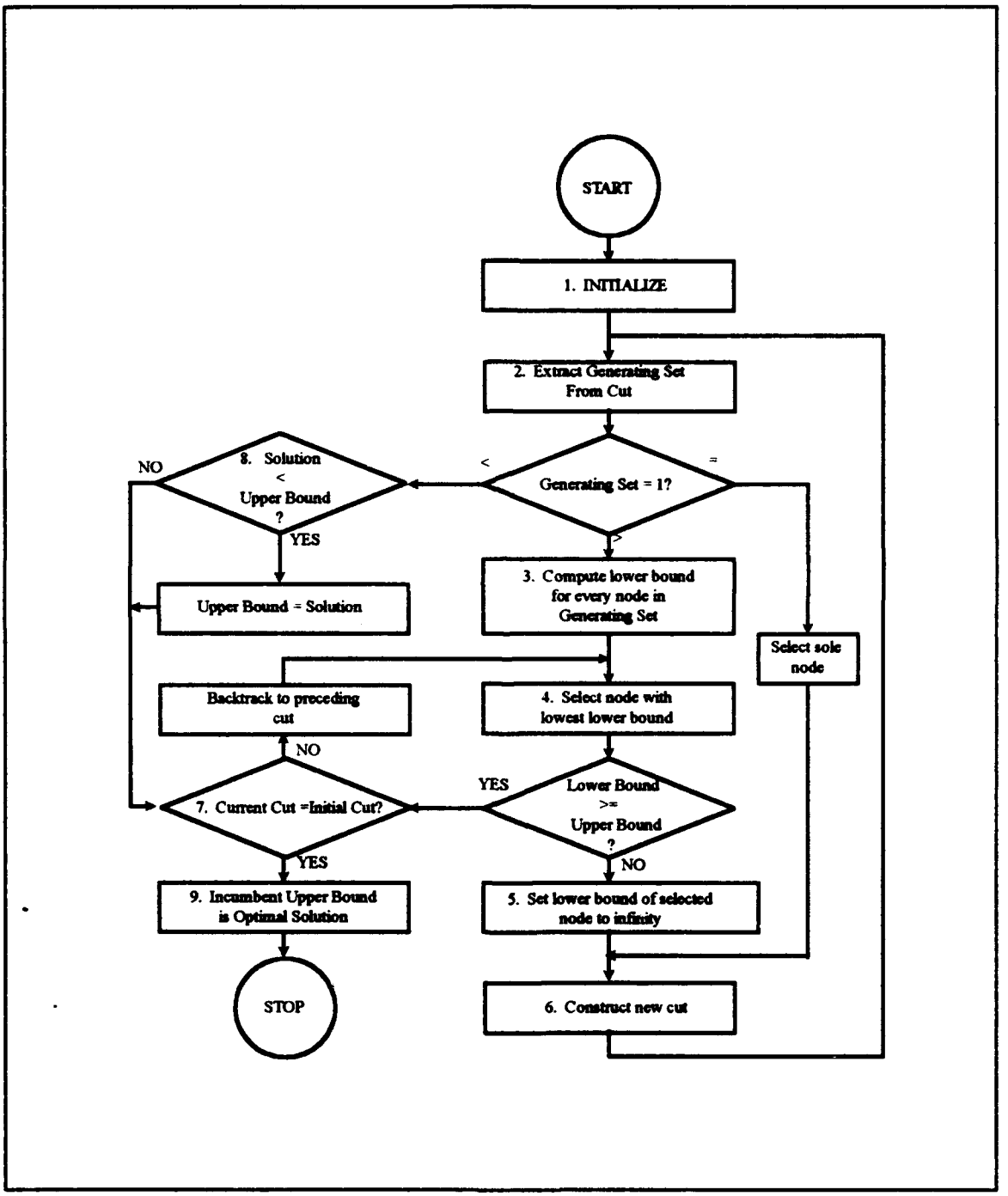


Figure 7: Algorithm Flowchart (adapted from Florian, Trepant, and McMahon, 1971:B-788)

Adaptation of the Basic Algorithm to a TOC Job Shop

Three basic changes were made to adapt the previously defined algorithm to a TOC job shop and allow a valid comparison to results generated by *DISASTER™*.

Constraints. *DISASTER™* only creates a schedule for constraint resources, and it controls all non-constraint resources through subordination. In order to provide schedules comparable with those generated by *DISASTER™* we ignored all non-constraint operations, and built a schedule for the constraint operations only. Subordination of non-constraint operations to our constraint schedule is beyond the scope of this research effort.

Shipping Buffer. *DISASTER™* defines a job as tardy if the last constraint operation is completed less than one-half of a shipping buffer before the due date of the job. If the operation is completed after this point, *DISASTER™* adds one full shipping buffer to the completion time of the last constraint operation to arrive at the anticipated completion time of the job. To allow a comparison with results generated by *DISASTER™*, we have used the same method to compute completion times.

Rods. Since we are only scheduling constraints, TOC mandates the use of rods between operations. In order to introduce this concept into our algorithm, we calculate an adjusted ready time for each operation. This new ready time is the earliest time that an operation can be scheduled without violating a rod. To allow a comparison with results generated by *DISASTER™*, we used the same half-buffer rod length, and the same method of calculating rod placement.

Model Coding

The algorithm was encoded using *Borland Turbo Pascal for Windows 1.5*. The program was compiled and run on a *Unisys* personal computer (IBM compatible) with a 386DX 33MHz processor. This machine had eight mega-bytes of RAM and a 387 coprocessor. The operating environment was *Microsoft DOS 5.0* with *Microsoft Windows 3.1*. A copy of the program file is included in Appendix B.

Experimental Factors

In accordance with our fourth research objective, a split-plot experimental design was created to manipulate several job shop characteristics. The performance of our algorithm was examined through the manipulation of three factors to observe the effect on a set of dependent variables. This experimental design was developed by Captain Stewart W. James and Captain Bruno A. Mediate in thesis AFIT/GSM/LAS/93S-9 to produce a set of benchmark problems and *DISASTER*™ solutions. A description of each factor and its levels are provided below.

Plant Type. The first experimental factor differentiates among three basic plant types used in this research effort: *A-plants*, *T-plants*, and *V-plants*. A single configuration of each plant type was used in the benchmark problems.

An *A-plant* is a plant in which operations converge, with a few finished products created from many raw materials (Fawcett and Pearson, 1991:50). For the purpose of this research effort, an A-plant will create one finished good from nine raw materials. A diagram of the A-plant used in this research is provided in Appendix A.

A *T-plant* is a plant in which several finished goods are created from several raw materials. This type of plant is usually an assemble-to-order operation with a number of common parts used to assemble the finished products (Fawcett and Pearson, 1991:51). For the purpose of this research effort, a T-plant will create five finished goods from four raw materials. A diagram of the T-plant used in this research is provided in Appendix A

A *V-plant* is a plant in which operations diverge, with many finished products created from a few raw materials (Fawcett and Pearson, 1991:50). For the purpose of this research effort, a V-plant will create five finished goods from one raw material. A diagram of the V-plant used in this research is provided in Appendix A.

Plants exhibiting characteristics of more than one of these plant types are called *combination plants* (Fawcett and Pearson, 1991:50). Combination plants will not be addressed in this research effort.

Percent Resource Criticality Factor. Of the ten resource types used in the previously defined plants, Captains James and Mediate defined two constraints, the *Blue* resource and the *Gold* resource. They operationalized the demand for each constraint as a percent resource criticality factor (%RCF). %RCF is defined as the ratio of total time a resource is required for a given time period to the total capacity available for that resource (Gargeya, 1992: 3). The second experimental factor expresses demand for the lower-constrained resource in terms of %RCF at three different levels: 105%, 115%, and 125%. These levels of %RCF are achieved in the

benchmark problems by varying the required processing time of each operation on the constraints, as depicted in Appendix A.

Percent Delta Resource Criticality Factor. Differences in demand between the two constraint resources is expressed in terms of their third experimental factor (%ΔRCF). This factor is expressed at three different levels: 0%, 25%, and 50%. These levels of %ΔRCF are achieved in the benchmark problems by varying the relative processing time of each operation on the constraints, as depicted in Appendix A.

Background Variables. Captains James and Mediate identified sixteen background variables, all but two of which were controlled by holding them constant. The exceptions were product types, and job order due dates.

Product Types. To achieve a wide range of conditions, a series of product types were defined, using various constraint process sequences. The following table identifies the constraint sequence for each plant type's products.

Table 3: Product Type Constraint Sequence (James and Mediate, 1993:115-141)

Plant Type	Product Type	Constraint Sequence
A	FG-D	Blue-Gold-Blue
T	FG-B	Blue-Blue-Gold-Blue
T	FG-C	Blue-Blue-Gold
T	FG-D	Gold
T	FG-F	Gold
T	FG-G	Blue-Blue-Gold-Gold
V	FG-A	Blue-Blue
V	FG-B	Blue-Gold
V	FG-D	Blue
V	FG-E	Gold-Gold
V	FG-G	Gold-Gold-Blue

Due Dates. Job order due dates were assigned according to a due date assignment rule which is based on the job order's arrival date, and the number of processing stations for the corresponding product type (Ragatz and Mabert, 1984:29). There were ten job orders for each replication. The product types were drawn from the table above, and the arrival dates were randomly selected without replacement from a two week period. This process was used to generate four replications for each treatment of the experimental factors, resulting in 108 benchmark problems (3 plant types x 3 %RCF levels x 3 %ΔRCF levels x 4 replications = 108 replications). The replications for each plant type were not identical, hence the split-plot experimental design. Details on the levels of the experimental factors for each replication are included in Appendix A. Since the problems involved two constraints, Captains James and Mediate ran each replication through *DISASTER*™ twice, reversing the sequence of the constraint schedule and thereby producing two separate schedules for each replication. Because of the simultaneous scheduling nature of our algorithm, we produced only one schedule for each replication.

Data Collection

We used the 108 problems developed by Captains James and Mediate to create an input data file for our algorithm. Running our program with this input file, we attempted to produce an optimal solution for each problem. We discovered that several problems would not produce a solution within a reasonable amount of processing time. As a result, we terminated processing for each problem after 5

minutes of CPU time if an optimal solution had not yet been verified. At that point, we designated the incumbent solution as a "heuristic" solution. These solutions may, in fact, have been optimal, but the time required to verify the optimality of the solution exceeded the designated time limit.

Statistical Analysis

In order to satisfy research objectives three and four, we performed a series of statistical comparisons using descriptive statistics and analysis of variance (ANOVA). All analyses were conducted using the *STATISTIX*™ 4.0 software package.

Statistical Comparisons. In order to compare the maximum tardiness of schedules created by *DISASTER*™ with that of schedules created by a simultaneous scheduling method (research objective three), we must compare the samples of data collected using both solution methods. Tests for statistical comparisons among the means or medians of two samples of data (i.e. paired t-test, sign test) require independence of the data within each sample set. Our split-plot experimental design introduces dependence among these values, since the same four replications are repeated within each setting for plant type (this is a characteristic of the split plot design). Therefore, statistical tests of the means or medians of these samples are not valid (Reynolds, 1993). As a result, we analyzed all comparisons between the two solution methods of data qualitatively.

Analysis of Variance. Our fourth research objective requires an investigation of the extent to which the performance of a simultaneous scheduling method is related to

the characteristics of the job shop (research objective four), an ANOVA was used to examine the variance of the mean values for dependent variables resulting from different levels of each of the experimental factors. The model used for a three-factor split-plot ANOVA for the dependent variable (X_{ijkl}) is defined below, as adapted from Kirk, 1982.

$$X_{ijkl}=\mu+\alpha_j+\pi_{i(j)}+\beta_k+(\alpha\beta)_{jk}+(\beta\pi)_{ki(j)}+\gamma_l+(\alpha\gamma)_{jl}+(\gamma\pi)_{li(j)}+(\beta\gamma)_{kl}+(\alpha\beta\gamma)_{jkl}+(\beta\gamma\pi)_{kli(j)}+\epsilon_{ijkl} \quad (16)$$

when: μ_{ijkl}= mean estimate of X

$\alpha_j,\beta_k,\gamma_l$= main effects parameters

$\pi_{i(j)}$=confounding interaction error parameter

$(\alpha\beta)_{jk},(\alpha\gamma)_{jl},(\beta\gamma)_{kl}$= two-factor interactions

$(\beta\pi)_{ki(j)},(\gamma\pi)_{li(j)},$= two-factor error parameters

$(\alpha\beta\gamma)_{jkl}$= three-factor interactions

$(\beta\gamma\pi)_{kli(j)}$= three-factor error parameter

ε_{ijkl}= random effects error

and: i = number of replication =1..4

j = values for PLANT=A, V, T

k = values for %RCF=105, 125, 150

l = values for %ΔRCF=0, 25, 50

Our null hypothesis (H_O) was that the values of μ_{ijk} are equal in the ANOVA model. Our alternate hypothesis (H_A) was that the μ_{ijk} terms are different.

The ANOVA of the split-plot model accounts for the dependence introduced by the presence of the confounding variable (replications) and the resulting error parameters shown above. However, an analysis of differences in the results generated by our algorithm and those generated by *DISASTER*™ among the levels of experimental factors could not be examined through the split-plot ANOVA. This comparison would have to account for another set of treatments introduced by the two solution procedures, creating an environment that must be addressed by a split-split plot ANOVA (Reynolds, 1993). The execution this analysis is highly complex, and is beyond the scope of this research effort. Consequently, ANOVA will be used when addressing research objective four, but not research objective three.

Assumptions of the ANOVA Model. There are three common assumptions for standard ANOVA models: independence among groups, normality of the sampling distributions of sample means, and homogeneity of variance. As previously mentioned, the condition of independence among groups does not apply in experimental situations in which different treatments are applied to the same subject (a split-plot design) (Cody and Smith, 1991: 136). The condition of normality is easily verified through the use of a Wilk-Shapiro test. All of our sampling distributions demonstrated a Wilk-Shapiro statistic of greater than 0.9, indicating good normality (Reynolds, 1993). All of our sample distributions also exhibit good homogeneity of variance when examined via a scatterplot.

Post-ANOVA Analysis. Statistically significant differences among the means associated with different levels of the three experimental factors were investigated

using a Tukey Multiple Comparisons Method (p=0.05). This test is used to rank the means and test for pair-wise differences among the means (Freund and Littell, 1981:57). Sample means that differ by more than the critical value (calculated using the appropriate degrees of freedom and levels of significance) are found to be statistically different from the others (Freund and Littell, 1981:56).

The ANOVA also identifies interaction among the experimental factors. Any interaction with a p-value greater than 0.05 was examined by plotting the means associated with each level of the experimental factors. The interaction plots were visually inspected to locate cross-over points where the values of one experimental factor demonstrated different trends given the values of the other experimental factors. This plot makes it possible to identify the response of the dependent variable to combinations of the experimental factors that may have been nullified in the ANOVA.

Summary

In this chapter, we examined our basic algorithm for scheduling a job shop to minimize maximum tardiness. This algorithm required several modifications to incorporate TOC and allow valid comparisons with *DISASTER*™. We reviewed these modifications, along with our experimental design and data collection procedures. Finally, we presented our method for statistical analysis.

IV. Results

We used several statistical comparisons to analyze the results of our research effort. The primary purpose of these comparisons was to address our third and fourth research objectives by comparing the maximum tardiness of schedules created by *DISASTER*™ with those created by a simultaneous scheduling method, and determining the extent to which the performance of a simultaneous scheduling method is related to characteristics of the job shop. The data used for these analyses can be found in tabular form in Appendices C and D.

Comparison with DISASTER™ Results

In order to asses the maximum tardiness of schedules created by *DISASTER*™ with those created by a simultaneous scheduling method (research objective three), we conducted three sets of comparisons. The data groups used for these comparisons are outlined in Table 4. Details of each comparison follow the table.

Table 4: Data Grouping for Comparison of Schedules

GROUP	DEFINITION
ALLCASE	Best solution (optimal or heuristic) found by our algorithm within the five minute time limit for maximum number of days tardy for each problem
OPTCASE	Optimal solution found by our algorithm for maximum number of days tardy for each problem
$DISASTER_{best}$	Best solution for maximum number of days tardy for each problem as determined by *DISASTER*™
$DISASTER_{worst}$	Worst solution for maximum number of days tardy for each problem as determined by *DISASTER*™
$DISASTER_{int}$	Best solution for maximum number of days tardy for each problem as determined by *DISASTER*™ (excluding problems without constraint interaction)
$\%\Delta TARDY_{best}$	Percent reduction in maximum tardiness for each problem between our best solution (optimal or heuristic) and *DISASTER*™'s best solution (with or without interactive constraints) $$\%\Delta TARDY_{best} = \frac{(DISASTER_{best} - ALLCASE)}{DISASTER_{best}} \quad (17)$$
$\%\Delta TARDY_{worst}$	Percent reduction in maximum tardiness for each problem between our best solution (optimal or heuristic) and *DISASTER*™'s worst solution (with or without interactive constraints) $$\%\Delta TARDY_{worst} = \frac{(DISASTER_{worst} - ALLCASE)}{DISASTER_{worst}} \quad (18)$$
$\%\Delta TARDY_{int}$	Percent reduction in maximum tardiness for each problem between our optimal solution and *DISASTER*™'s best solution with interactive constraints. $$\%\Delta TARDY_{int} = \frac{(DISASTER_{int} - ALLCASE)}{DISASTER_{int}} \quad (19)$$

Comparison with DISASTER™ Best Case. This comparison was made between the tardiness for each problem given the best solutions (optimal or heuristic) obtained using our algorithm within the five minute time limit (ALLCASE) and the tardiness for the same problems given the best solutions obtained for the same problems using

DISASTER™ (DISASTER$_{best}$). This comparison gives some indication of the degree to which *DISASTER*™ is capable of achieving the minimum possible maximum tardiness for a given problem. Descriptive statistics for each dependent variable, as well as the percent difference between the dependent variables (%ΔTARDY$_{best}$) are provided in Table 5.

Table 5: Descriptive Statistics - ALLCASE vs DISASTER$_{best}$

VARIABLE	MEAN TARDINESS	MEDIAN TARDINESS	STANDARD DEVIATION
ALLCASE	7.2 days	7.0 days	2.9 days
DISASTER$_{best}$	7.9 days	7.0 days	2.8 days
%ΔTARDY$_{best}$	2.2 %	0.0 %	10.2 %

A histogram of the percent differences (%ΔTARDY$_{best}$) is shown in Figure 8. Positive values reflect superior results by simultaneous scheduling of the constraints. Negative

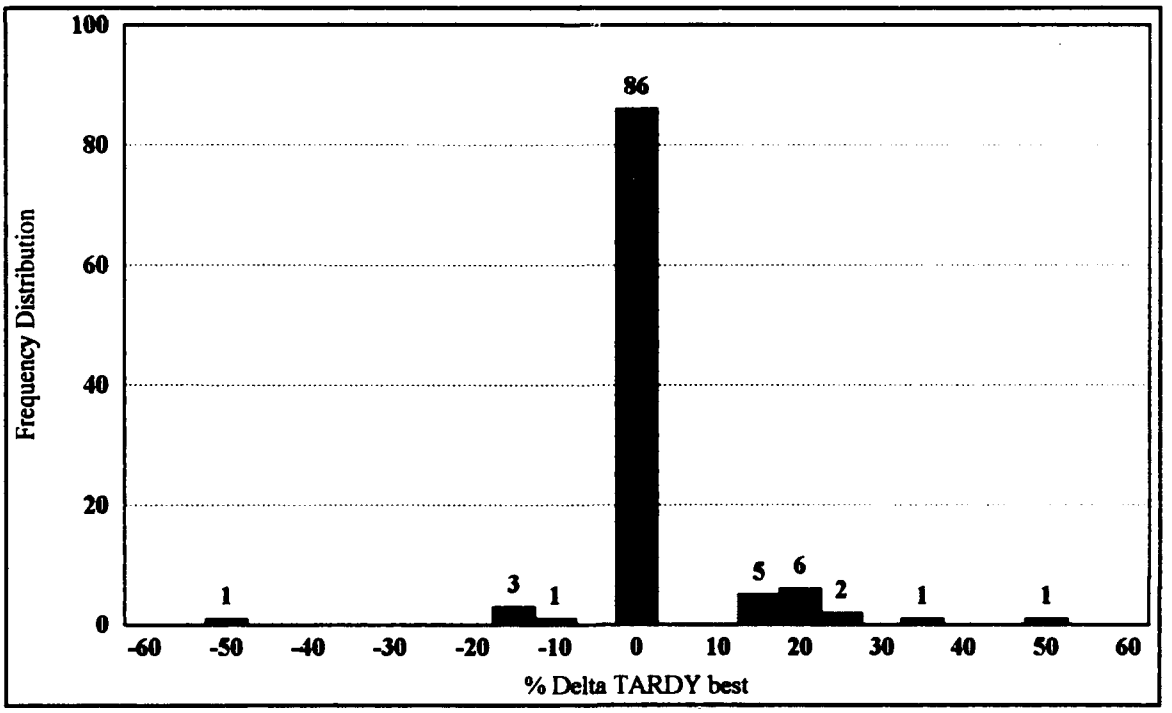


Figure 8: Histogram of %ΔTARDY$_{best}$

values do not represent superior performance by *DISASTER*™. In these cases, *DISASTER*™ identified only one constraint, and therefore solved a simpler problem than the one solved by our algorithm.

Inspection of the histogram revealed that although a one case resulted in a difference in minimized maximum tardiness as large of 50%, the majority of cases (76%) resulted in no difference. This observation, along with the descriptive statistics indicated that $DISASTER_{best}$ typically performed as well as ALLCASE.

Comparison with DISASTER™ Worst Case. This comparison was made between the tardiness for each problem given the best solutions (optimal or heuristic) obtained using our algorithm within the five minute time limit (ALLCASE) and the tardiness for the same problems given the worst solutions obtained for the same problems using *DISASTER*™ ($DISASTER_{worst}$). This comparison gives some indication of the degree to which a user of *DISASTER*™ may fail to achieve the minimum possible maximum tardiness for a given problem if they fail to schedule constraints in the best sequence. Descriptive statistics for each dependent variable, as well as the percent difference between the dependent variables ($\%\Delta TARDY_{worst}$) are provided in Table 6.

Table 6: Descriptive Statistics - ALLCASE vs $DISASTER_{worst}$

VARIABLE	MEAN TARDINESS	MEDIAN TARDINESS	STANDARD DEVIATION
ALLCASE	7.2 days	.7.0 days	2.9 days
$DISASTER_{worst}$	9.1 days	9.0 days	3.5 days
$\%\Delta TARDY_{worst}$	19.3 %	15.5 %	18.0 %

A histogram of the percent differences ($\%\Delta TARDY_{worst}$) is shown in Figure 9.

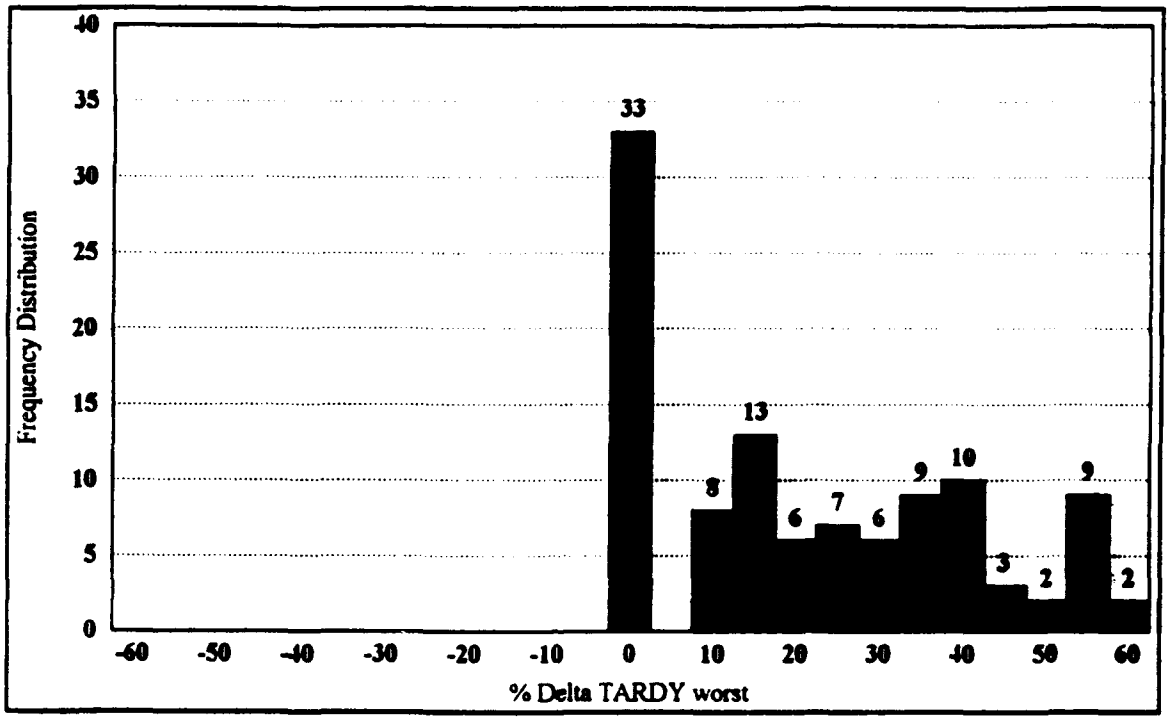


Figure 9: Histogram of $\%\Delta TARDY_{worst}$

Inspection of the histogram revealed a difference in the majority (70%) of cases. For many cases this difference was great, as high as 60%. This observation, along with the descriptive statistics indicated that ALLCASE performance was typically and substantially better than that of $DISASTER_{worst}$.

Comparison of Optimal and Valid Cases. Out of the 108 benchmark problems provided by Captains James and Mediate, 39 did not provide a satisfactory solution for comparison. In these 39 problems, the best case solution provided by *DISASTER™* failed to account for the existence of the second constraint, thereby eliminating constraint interaction. Additionally, our algorithm was unable to find and verify an optimal solution for 26 of the benchmark problems. Therefore, we limited our third comparison to the 51 problems for which both an optimal solution was obtained using

our algorithm (OPTCASE), and interactive constraints existed in the solution provided by *DISASTER*™ ($DISASTER_{int}$). This comparison identifies the differences in the best solutions provided by *DISASTER*™ and the optimal value for minimized maximum tardiness for a given problem, without any of the complicating factors mentioned above. Descriptive statistics for each dependent variable, as well as the percent difference between the dependent variables ($\%\Delta TARDY_{int}$) are provided in Table 7. A histogram of percent differences ($\%\Delta TARDY_{int}$) is shown in Figure 10.

Table 7: Descriptive Statistics - OPTCASE vs $DISASTER_{int}$

VARIABLE	MEAN TARDINESS	MEDIAN TARDINESS	STANDARD DEVIATION
OPTCASE	7.2 days	7.0 days	2.9 days
$DISASTER_{int}$	7.9 days	8.0 days	3.1 days
$\%\Delta TARDY_{int}$	2.9 %	0.0 %	6.5 %

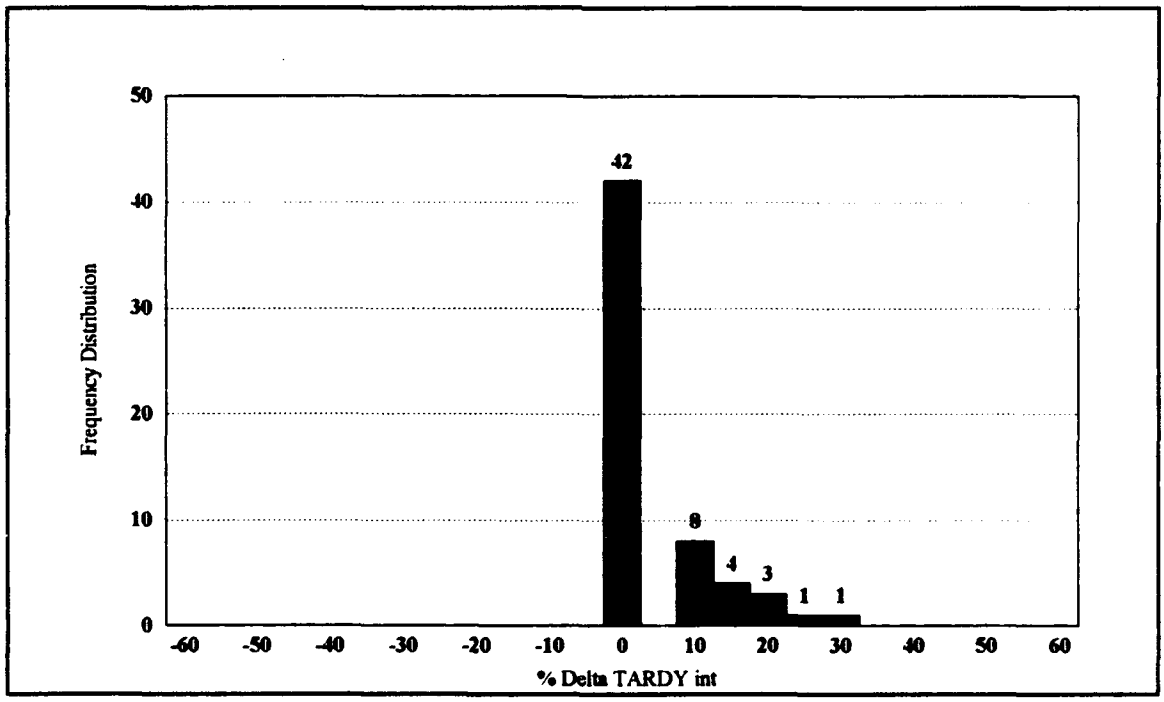


Figure 10: Histogram of $\%\Delta TARDY_{int}$

Inspection of the histogram revealed that the majority of cases (71%) resulted in no difference. This observation, along with the descriptive statistics indicated that although *DISASTER*™ did not always achieve the optimal solution for minimized maximum tardiness, there were only occasional moderate differences between the values for this performance measure as represented by the variables OPTCASE and $DISASTER_{best}$.

Investigation of Algorithm Performance

In order to determine the extent to which the performance of a simultaneous scheduling method is related to the characteristics of the job shop (research objective four), we examined the relationship between the performance of our algorithm and the characteristics of the job shop designated as factors in our experimental design. These factors are listed in Table 8.

Table 8: Experimental Factors

FACTOR	DEFINITION	LEVELS
PLANT	Job shop configuration	A, T, V
%RCF	Percent resource criticality factor	105, 115, 125
%ΔRCF	Difference in percent resource criticality factor	0, 25, 50

For each problem, we recorded two parameters which characterized the performance of our algorithm. First, we recorded the time required to find the best solution (without verification) for our objective of minimized maximum tardiness (T_{FIND}). This measure gives an indication of the strength of our branching technique.

Next, we recorded the time required to verify the best solution (exclusive of the time to find) for our objective of minimized maximum tardiness (T_{VERIFY}). This measure gives an indication of the strength of our lower bound. If the best solution was not verified within five minutes (300 seconds), then verification time was designated as

$$T_{VERIFY} = 300 - T_{FIND} \qquad (20)$$

For the 26 problems where the best solution was not verified as optimal within five minutes, T_{VERIFY} represents an optimistic measure of the actual time required to verify our best solution. Therefore, the statistical significance of any differences in our values for T_{VERIFY} across the various levels of the experimental factors are likely to be understated in this analysis.

We analyzed both dependent parameters with respect to the different levels of the factors of the experimental design using a split-plot ANOVA. The ANOVA indicated no in difference T_{FIND} across the different levels of the experimental factors. However, the ANOVA did indicate a significant difference in T_{VERIFY} across different levels of PLANT (P-value = 0.0004). A Tukey test showed that our algorithm verified the optimality of the best solution for our objective function in problems with PLANT = V significantly faster than in problems with PLANT = A or T. There were no significant differences across th' fferent levels of the other two experimental factors.

Summary

In this chapter, we presented the results of our analyses in two parts. First, we compared the schedules generated by our algorithm with those generated by *DISASTER*™. We demonstrated that with proper constraint sequencing, *DISASTER*™ is capable of achieving a level of maximum tardiness similar to that achieved by our algorithm. However, we also showed that if the constraint sequencing in *DISASTER*™ is not chosen carefully, our algorithm was able to achieve lower maximum tardiness.

Next, we investigated the solution characteristics of our algorithm. We found that there was no difference in time to find the best solution across the different levels of the experimental factors, but that there was a significant difference in time to verify the best solution across different values of PLANT, with V-Plant problems being verified significantly faster than T-Plant or A-Plant problems.

Table 9: Summary of Results

COMPARISON	RESULTS
ALLCASE vs $DISASTER_{best}$	Mean difference = 2.2% Median difference = 0.0% Standard deviation of difference = 10.2%
ALLCASE vs $DISASTER_{worst}$	Mean difference = 19.3% Median difference = 15.5% Standard deviation of difference = 18.0%
OPTCASE vs $DISASTER_{int}$	Mean difference = 2.9% Median difference = 0.0% Standard deviation of difference = 6.5%
T_{FIND}	No significant difference across levels of the experimental factors
T_{VERIFY}	Faster (P-value = 0.0004) verification in problems with PLANT = V vs problems with PLANT = A or V

V. Conclusions and Recommendations

In the previous chapter, we presented the results of our research. In this chapter we will examine these findings as they relate to our research objectives, and offer some suggestions for further research.

Problem Formulation and Solution Methodology

We formulated the job shop problem for the objective of minimized maximum tardiness in a manner consistent with the principles of TOC. This formulation differed from traditional job shop formulations primarily due to the introduction of rods. The presence of rods was reflected by dynamic ready times among the constraint operations. We were unable to identify an optimal solution technique consistent with our formulation during our review of the literature. We therefore modified the Graph-Theoretical method of Florian, Trepant, and McMahon to reflect the performance measure of minimized maximum tardiness and the relevant concepts of TOC.

Comparison with DISASTER™ Results

In general, our algorithm demonstrated little improvement over *DISASTER™*'s best performance for the objective of minimizing maximum tardiness. A mean reduction in tardiness of 2.9% was achieved between *DISASTER™*'s best solutions for problems with interacting constraints and our optimal solutions for the same problems.

In 71% of these problems, there was little or no difference between the maximum tardiness of schedules generated by our algorithm and those created by *DISASTER*™. This improvement was less than we initially expected since our algorithm was designed solely to find an optimal schedule to minimize maximum tardiness, while *DISASTER*™'s heuristic solution technique attempts to meet numerous scheduling objectives. Our results indicate that *DISASTER*™'s heuristic solution technique is very strong for this performance measure.

Despite the excellent performance of *DISASTER*™'s best solutions there is still room for improvement. In 29% of the problems comparing our optimal solutions and *DISASTER*™'s solutions with interacting constraints, we achieved at least 10% better performance for the objective of minimizing maximum tardiness. The improvement demonstrates that simultaneous scheduling of the constraints can result in better schedule performance, which may translate into increased system throughput.

Our algorithm demonstrated significant improvement (mean = 19.3%) over *DISASTER*™'s worst performance even when heuristically limited to five minutes of processing. It was also observed that the sequence of constraints resulting in the worst solution for minimized maximum tardiness was occasionally the sequence which was recommended by *DISASTER*™. Our improvement over *DISASTER*™'s worst performance suggests that users of *DISASTER*™ should examine the results of all possible constraint sequences to find the best schedule that *DISASTER*™ is capable of producing for a given performance measure.

Investigation of Algorithm Performance

We determined that there was no difference across the various levels of the experimental factors in the time required to find the best solution for our objective of minimized maximum tardiness. However, we did find a statistically significant difference across levels of the experimental factors in the time required to verify the best solution for our performance objective. Solutions to problems in V-plants were verified significantly faster than solutions to problems in T-plants and A-plants.

There are a greater number of operations in A-plants and T-plants than V-plants. In order to maintain comparable total constraint loading, the problems with more operations have shorter processing times per operation. The rods create a greater interval between operations as the processing time per operation decreases. Our bound does not account for the presence of rods. Thus, it appears that our bound is weaker in plants with shorter processing times because the best achievable tardiness is significantly greater (due to the gaps in the schedule) than that estimated by simple early-due-date sequencing, which ignores the presence of rods.

Recommendations For Further Research

Scheduling practices using TOC have not been subjected to a great deal of empirical research. As such, there has been little information available to evaluate the results of this process in a controlled environment. This research has demonstrated the relative strength of *DISASTER*™'s heuristic constraint scheduling technique for the objective of minimized maximum tardiness. Continued evaluation of this heuristic is

encouraged, since this portion of the software is a crucial to determining the throughput of the production system. Any increase in throughput as a result of improvements to this heuristic or other aspects of *DISASTER*™ will be very valuable to users of this software.

There are three general areas where we identified a need for further research. First, there are several areas in which our algorithm could be improved. A new lower bound could be developed which is able to consider the effects of rods. The utility of the algorithm could also be enhanced by addressing other parts of TOC scheduling such as subordination of non-constraints to the constraint schedule. Also, heuristic solution techniques could be developed which avoid lengthy optimality verification. These improvements would create a viable scheduling alterative to *DISASTER*™. This alternative would not be dependent on proper constraint sequencing for maximum performance.

Another area rich in opportunity is further evaluation of *DISASTER*™. Algorithms should be developed to evaluate the ability of *DISASTER*™ to maximize or minimize another objective - number of late jobs, makespan, etc. This would reflect the other management concerns relevant to job shop scheduling. Additionally, it would be very useful to develop an ability to predict the small percentage of cases where *DISASTER*™'s solutions are far from optimal and devise heuristic adjustment procedures to deal with those cases.

The evaluation of *DISASTER*™ could also be enhanced by the development of a new experimental design with compete independence among the benchmark problems.

A more thorough analysis could then be conducted to determine the relationship between the optimality of *DISASTER*™'s solutions and the characteristics of the job shop.

The final area of further research opportunity which we envision is in the application of TOC scheduling to the USAF. Multiple constraints are known to exist in many USAF operations. The budget-cutting activities of 1990's can be expected to put more demand on our remaining resources, which will increase the likelihood of encountering system constraints. An empirical study of USAF systems is needed to identify the number and nature of existing constraints in USAF systems (i.e. activities at an Air Force Logistics Center). This research will help focus the efforts of both future users and researchers of TOC.

Conclusion

This research effort has provided the first empirical evidence of the effectiveness of the *DISASTER*™ scheduling program. The existing partnerships with the Avraham Y. Goldratt Institute will allow the Air Force to become an influential participant in the further refinement of *DISASTER*™. We can expect that the management of the constraints in USAF operations will contribute to future success in executing our global mission. As the Air Force is challenged to increase its effectiveness despite shrinking resources, the strategies expressed in TOC will be a valuable source of guidance for continuous improvement efforts.

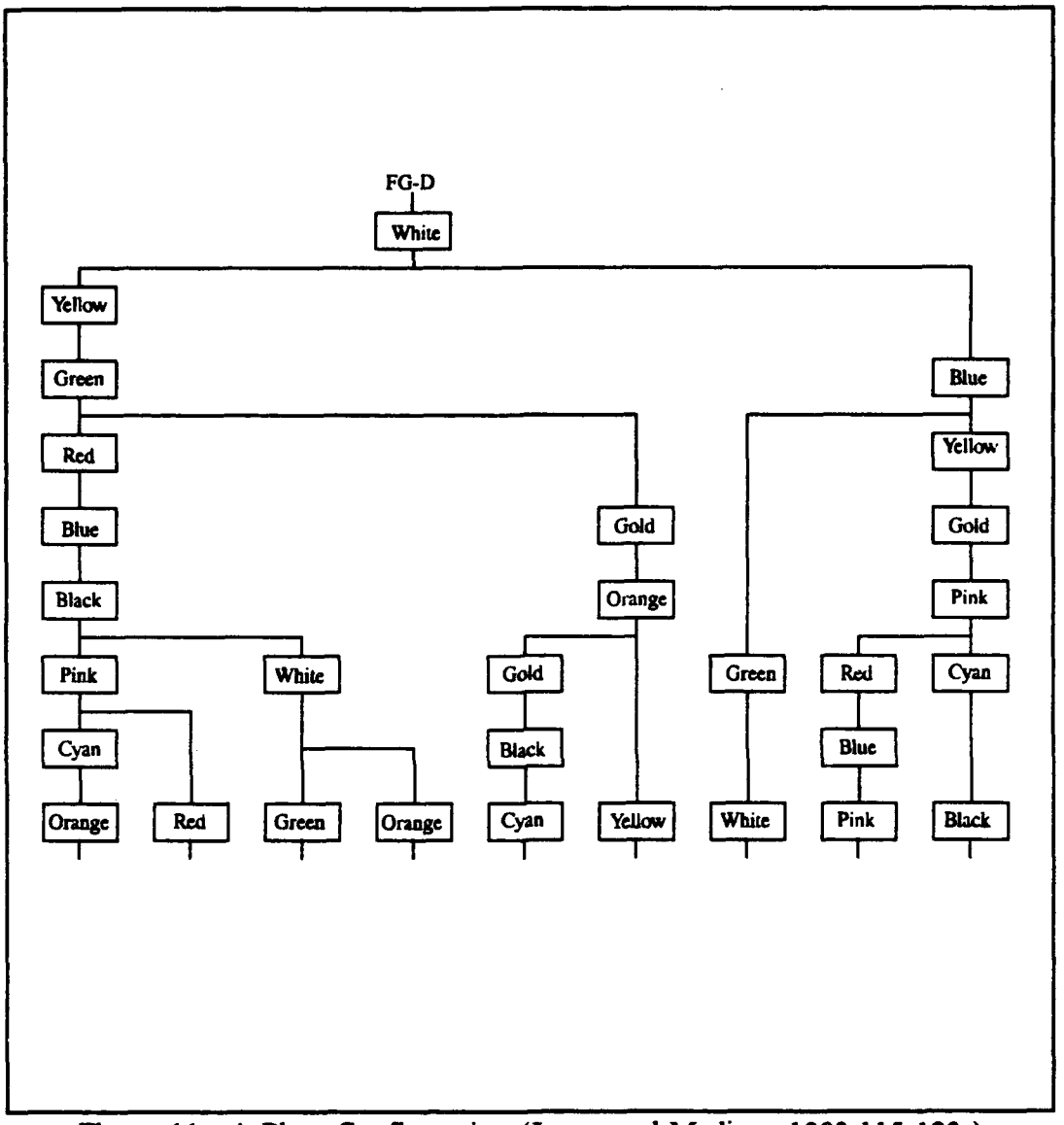


Figure 11: A-Plant Configuration (James and Mediate, 1993:115-123)

Table 10: A-Plant Constraint Loading (James and Mediate, 1993: 115-123)

%RCF	%ΔRCF	Blue Processing Time	Gold Processing Time
105	0	170	170
105	25	170	210
105	50	170	250
115	0	185	185
115	25	185	230
115	50	185	275
125	0	200	200
125	25	200	250
125	50	200	300

Table 11: A-Plant Job Order Due Dates (James and Mediate, 1993: 115-123)

Rep One		Rep Two		Rep Three		Rep Four	
Product	Due Date	Product	Due Date	Product	Due Date	Product	Due Date
FG-D	04 Oct 93	FG-D	04 Oct 93	FG-D	05 Oct 93	FG-D	05 Oct 93
FG-D	05 Oct 93	FG-D	07 Oct 93	FG-D	05 Oct 93	FG-D	06 Oct 93
FG-D	05 Oct 93	FG-D	07 Oct 93	FG-D	07 Oct 93	FG-D	07 Oct 93
FG-D	06 Oct 93	FG-D	08 Oct 93	FG-D	08 Oct 93	FG-D	08 Oct 93
FG-D	07 Oct 93	FG-D	12 Oct 93	FG-D	11 Oct 93	FG-D	12 Oct 93
FG-D	11 Oct 93	FG-D	13 Oct 93	FG-D	12 Oct 93	FG-D	13 Oct 93
FG-D	13 Oct 93	FG-D	14 Oct 93	FG-D	13 Oct 93	FG-D	13 Oct 93
FG-D	13 Oct 93	FG-D	14 Oct 93	FG-D	13 Oct 93	FG-D	15 Oct 93
FG-D	14 Oct 93	FG-D	15 Oct 93	FG-D	14 Oct 93	FG-D	15 Oct 93
FG-D	15 Oct 93	FG-D	15 Oct 93	FG-D	15 Oct 93	FG-D	15 Oct 93

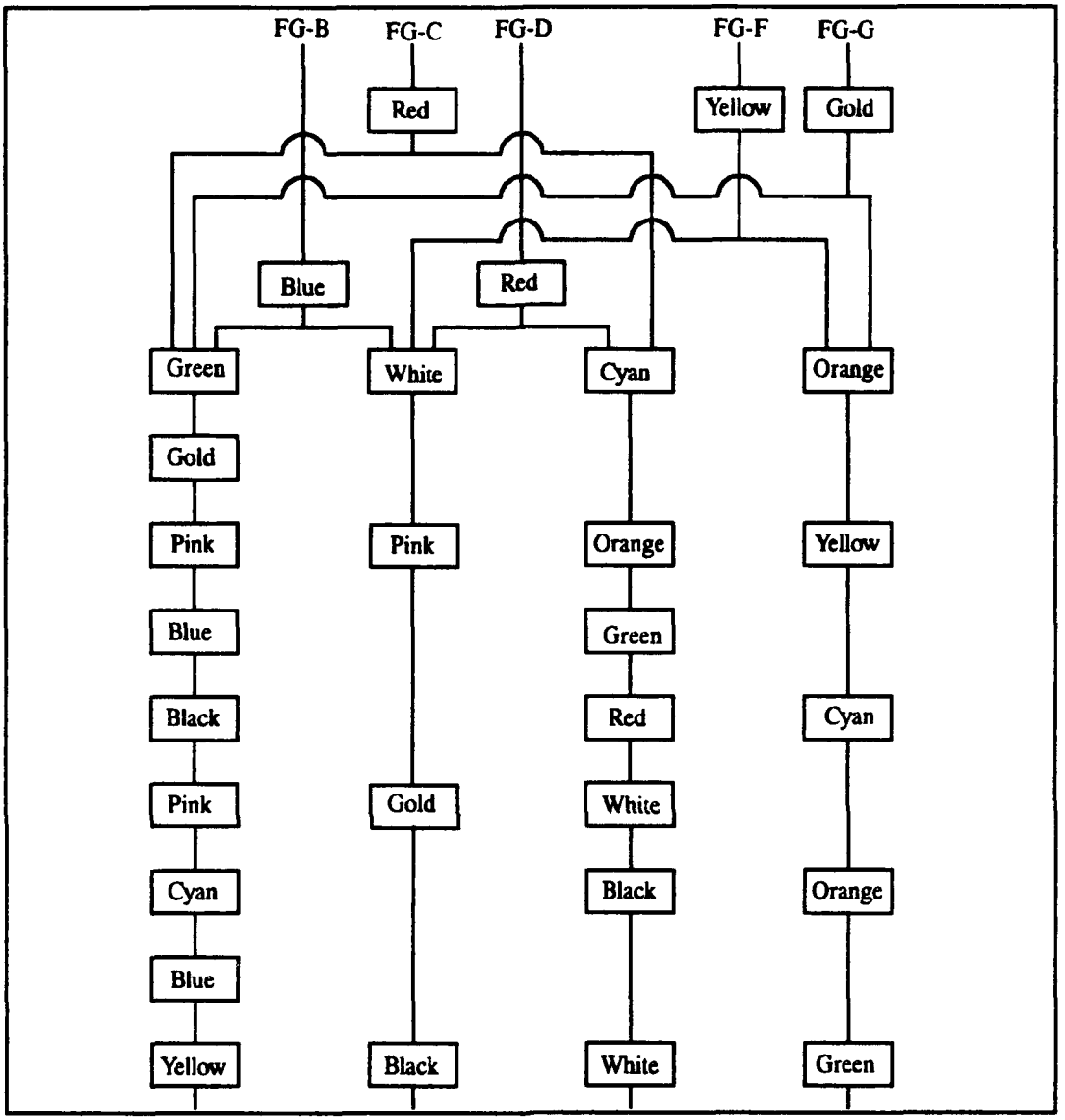


Figure 12: T-Plant Configuration (James and Mediate, 1993: 124-132)

Table 12: T-Plant Constraint Loading (James and Mediate, 1993:124-132)

%RCF	%ΔRCF	Blue Processing Time	Gold Processing Time
105	0	360	360
105	25	360	450
105	50	360	540
115	0	400	400
115	25	400	500
115	50	400	600
125	0	430	430
125	25	430	540
125	50	430	640

Table 13: T-Plant Job Order Due Dates (James and Mediate, 1993: 124-132)

Rep One		Rep Two		Rep Three		Rep Four	
Product	Due Date	Product	Due Date	Product	Due Date	Product	Due Date
FG-B	04 Oct 93	FG-D	04 Oct 93	FG-D	06 Oct 93	FG-C	04 Oct 93
FG-C	05 Oct 93	FG-G	05 Oct 93	FG-F	06 Oct 93	FG-D	04 Oct 93
FG-B	07 Oct 93	FG-C	05 Oct 93	FG-D	07 Oct 93	FG-F	06 Oct 93
FG-F	07 Oct 93	FG-G	06 Oct 93	FG-F	07 Oct 93	FG-D	08 Oct 93
FG-F	08 Oct 93	FG-B	07 Oct 93	FG-C	08 Oct 93	FG-F	11 Oct 93
FG-D	11 Oct 93	FG-F	07 Oct 93	FG-B	13 Oct 93	FG-G	11 Oct 93
FG-C	12 Oct 93	FG-F	08 Oct 93	FG-G	13 Oct 93	FG-G	13 Oct 93
FG-D	12 Oct 93	FG-B	11 Oct 93	FG-B	14 Oct 93	FG-B	14 Oct 93
FG-G	14 Oct 93	FG-D	11 Oct 93	FG-G	14 Oct 93	FG-B	14 Oct 93
FG-G	15 Oct 93	FG-C	15 Oct 93	FG-C	15 Oct 93	FG-C	15 Oct 93

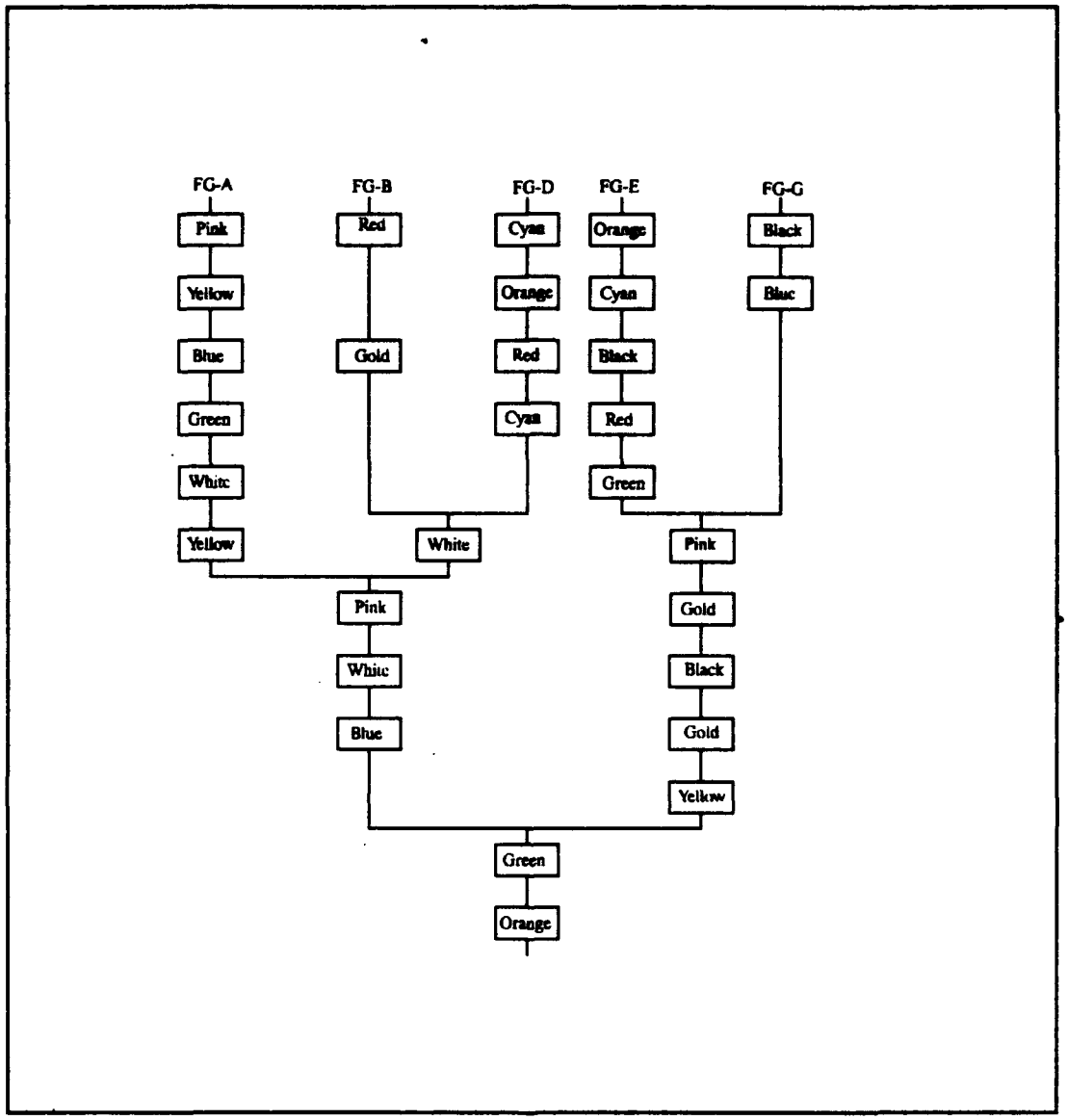


Figure 13: V-Plant Configuration (James and Mediate, 1993:133-141)

Table 14: V-Plant Constraint Loading (James and Mediate, 1993:133-141)

%RCF	%ΔRCF	Blue Processing Time	Gold Processing Time
105	0	500	500
105	25	500	650
105	50	500	750
115	0	550	550
115	25	550	690
115	50	550	830
125	0	600	600
125	25	600	750
125	50	600	750

Table 15: V-Plant Job Order Due Dates (James and Mediate, 1993:133-141)

Rep One		Rep Two		Rep Three		Rep Four	
Product	Due Date	Product	Due Date	Product	Due Date	Product	Due Date
FG-G	04 Oct 93	FG-E	04 Oct 93	FG-G	04 Oct 93	FG-A	04 Oct 93
FG-A	07 Oct 93	FG-A	06 Oct 93	FG-B	05 Oct 93	FG-A	05 Oct 93
FG-A	08 Oct 93	FG-B	06 Oct 93	FG-D	06 Oct 93	FG-G	05 Oct 93
FG-B	08 Oct 93	FG-D	07 Oct 93	FG-A	12 Oct 93	FG-B	06 Oct 93
FG-E	08 Oct 93	FG-G	08 Oct 93	FG-B	12 Oct 93	FG-E	06 Oct 93
FG-E	11 Oct 93	FG-B	12 Oct 93	FG-D	12 Oct 93	FG-G	06 Oct 93
FG-G	11 Oct 93	FG-D	12 Oct 93	FG-G	12 Oct 93	FG-B	07 Oct 93
FG-B	12 Oct 93	FG-G	12 Oct 93	FG-A	15 Oct 93	FG-E	07 Oct 93
FG-D	14 Oct 93	FG-A	13 Oct 93	FG-E	15 Oct 93	FG-D	08 Oct 93
FG-D	15 Oct 93	FG-E	15 Oct 93	FG-E	15 Oct 93	FG-D	15 Oct 93

Appendix B: Software Code Listing

```
program Thesis;

uses
 WinCrt, WinDos, WinProcs;

const
  m = 40;
  numjob = 10;
  bignum = 32000;

type
  rvector = array[0..61] of real;
  vector =  array[0..61] of integer;
  dblvector = array[0..61,1..2] of integer;
  check =  array[0..61] of boolean;
  matrix =  array[0..61,0..m] of integer;
  rmatrix = array[0..61,0..m] of real;
  dual = array[1..2] of real;

var
  space, checktime, solvetime, starttime, finishtime, mtard, upbound, temp:real;
  mlowbound, jlowbound, lowbound, tempbound, nodecnt: real;
  a, j, i, n, k, filend, branch, level, select, rcf, drcf, genset, rep, termnode: integer;
  overdue, start, finish: rvector;
  job, machine, gensize, duration, due, gold, blue: vector;
  pred: dblvector;
  scheduled, cut: check;
  GC: matrix;
  GCbound: rmatrix;
  indata, outdata: text;
  done, solution, test: boolean;
  hour, minute, second, sec100: word;
  plant: char;
  machfin: dual;


{* NEWSTART PROCEDURE --Calculates new start times when an operation is scheduled*}
procedure Newstart(var nstart, nfinish: rvector; var nscheduled: check; var nselect: integer);

var
  mcnt, j, othermach : integer;
  tempstart: real;

begin
```

```
machfin[1] := 0;                    {Initialize machine one finish time}
machfin[2] := 0;                    {Initialize machine two finish time}

if machine[nselect] = 1             {Determine machine for selected operation}
  then othermach := 2
  else othermach := 1;

for i := 1 to n do                                  {Set finish time of selected machine}
  if (machine[i] = machine[nselect])    {to the finish time of latest scheduled operation}
    and (nscheduled[i] = TRUE)
      and (nfinish[i] > machfin[machine[nselect]])
        then machfin[machine[nselect]] := nfinish[i];

for i := 1 to n do                  {Set finish time of non-selected machine}
  if (machine[i] = othermach)       {to finish time of the latest scheduled operation}
    and (nscheduled[i] = TRUE)
      and (nfinish[i] > machfin[othermach])
        then machfin[othermach] := nfinish[i];

for j := 1 to 2 do                  {Calculate inter-operation spacers}
  if pred[nselect,j] > 0 then
  begin
    space := 0;
    if machine[nselect] = machine[pred[nselect,j]]
      then
      begin
        if duration[pred[nselect,j]] < duration[nselect]
          then space := (240-(0.99*duration[pred[nselect,j]]))
          else space := (240-(0.99*duration[nselect]));
        if space < 0 then space := 0;
        if (nfinish[pred[nselect,j]]+space) > nstart[nselect]
          then nstart[nselect]:=(nfinish[pred[nselect,j]]+space);
      end
      else
      begin
        if duration[pred[nselect,j]] < duration[nselect]
          then space := (240-(0.99*duration[pred[nselect,j]]))
          else space := (240-(0.99*duration[nselect]));
        if (nfinish[pred[nselect,j]]+space) > nstart[nselect]
          then nstart[nselect]:=(nfinish[pred[nselect,j]]+space);
       end;
  end;

if nstart[nselect] < machfin[machine[nselect]] {Determine early start/finish for each operation}
then nstart[nselect] := machfin[machine[nselect]];
nfinish[nselect] := nstart[nselect] + duration[nselect];
```

```
machfin[machine[nselect]] := nfinish[nselect];
nscheduled[nselect] := TRUE;
end;


{* MBOUND FUNCTION --Calculates lower bound for selected operation*}function
MBound(bselect: integer; bstart, bfinish: rvector; bdue: vector; bscheduled: check): real;

var
 mcnt, complete, short: integer;
 done: boolean;
 mfinish, tardy, tardymax: real;

begin
newstart(bstart, bfinish, bscheduled, bselect);{Calculate new start times given selected
operation}
tardymax := 0;

for i := 1 to n do                    {Determine maximum tardiness for scheduled ops}
  if (bscheduled[i] = TRUE)
    and ((bfinish[i]-bdue[i]) > tardymax)
      then tardymax := (bfinish[i] - bdue[i]);

for mcnt := 1 to 2 do                 {For each machine do...}
begin
  mfinish := machfin[mcnt];
  done := FALSE;

  while done = FALSE do                       {While due date calculations are not done
do...}
  begin
    complete := bignum;
    for i := 1 to n do                {Find operation with earliest due date}
      if (machine[i] = mcnt)
        and (bscheduled[i] = FALSE)
          and (bdue[i] < complete)
            then
              begin
                complete := bdue[i];
                short := i;
              end;
    if complete = bignum
      then done := TRUE
      else
      begin
        mfinish := mfinish + duration[short];
```

```
      tardy := mfinish - bdue[short];   {Determine tardiness of selected operation}
      if tardy > tardymax
        then tardymax := tardy;         {Update maximum tardiness}
    end;
   bscheduled[short] := TRUE;                    {Eliminate early unscheduled due date from
search}
  end;

end;

mbound := tardymax;                             {Return largest machine based lower bound}
end;


{*** GENERATE PROCEDURE -- Calculates the generating set for the current cut ***}
procedure Generate(var glevel: integer; var gcut: check; var gGC: matrix);

var
  genmach: integer;
  minfin: real;

begin
minfin := bignum;                   {Select machine for generating set}
glevel := glevel + 1;               {Increment level number}

for i := 1 to n do
  if (gcut[i] = TRUE)
    and (finish[i]<minfin)
      then
        begin
          minfin := finish[i];      {Find operation with earliest finish time}
          genmach := machine[i];    {Select machine corresponding to operation}

        end;

genset := 0;
for i := 1 to n do
  if (gcut[i] = TRUE)               {Select operations in cut/on machine}
    and (machine[i] = genmach)
      then
        begin
          genset := genset + 1;     {Increment operations in generating set}
          gGC[glevel,genset] := i;  {Add operation to generating set}
        end;

gensize[glevel] := genset;          {Save size of generating set at this level}
```

```
if genset = 0
  then solution := TRUE
end;


{*** SOLVE PROCEDURE – Calculates schedule solution during backtracking mode ***}
procedure solve(slevel: integer; sstart, sfinish: rvector; sblue, sgold: vector;
                ssolution: boolean; scut, sscheduled: check; sGC: matrix; sGCbound: rmatrix);

var
  j, sselect, initbranch, z:  integer;

label 1;

begin;
solution := FALSE;                    {Assume no solution exits}
sselect := sGC[slevel,branch];        {Select the given operation}
initbranch := branch;
for i := (slevel) to n do             {For all levels below that of selected operation do }
begin
  sblue[i] := 0;                      {Reinitialize schedule information}
  sgold[i] := 0;
  for j := 1 to gensize[i] do
  begin
    sstart[sGC[i,j]] := 0;
    sfinish[sGC[i,j]] := (sstart[sGC[i,j]]+duration[sGC[i,j]]);
    scut[sGC[i,j]] := FALSE;
  end;
end;

newstart(sstart,sfinish,sscheduled,sselect);{Calculate start times}

if machine[sselect] = 1               {Schedule selected operation on proper machine}
  then sblue[slevel] := sselect
  else sgold[slevel] := sselect;

for i := 1 to n do                    {Select successor operation for next cut}
  if (sscheduled[pred[i,1]] = TRUE)
    and (sscheduled[pred[i,2]] = TRUE)
      and (sscheduled[i] = FALSE)
        then scut[i] := TRUE;

Generate(slevel,scut,sGC);            {Generate next Generating Set}

While (solution = FALSE) do           {Until a solution is reached do...}
begin
```

```
temp := bignum;

for a:= 1 to genset do
begin                                   {Find the operation with the smallest lower bound}
  lowbound := mbound(sGC[slevel,a],sstart,sfinish,due,sscheduled);
  sGCbound[slevel,a] := lowbound;
  if lowbound < temp
    then
    begin
      branch := a;
      temp := lowbound;
    end;
end;

While temp >= upbound do                {If lower bound exceeds upper bound then do }
begin
  genset := gensize[slevel];            {Reset schedule information at this level}
  for a := 1 to genset do
  begin
    sstart[sGC[slevel,a]] := 0;
    sfinish[sGC[slevel,a]] := (sstart[sGC[slevel,a]]+duration[sGC[slevel,a]]);
    sscheduled[sGC[slevel,a]] := FALSE;
    scut[sGC[slevel,a]] := FALSE;
  end;
  sblue[slevel] := 0;
  sgold[slevel] := 0;
  slevel := slevel - 1;                 {Backtrack one level}
  if slevel <= level                    {If backtracked to subroutine entry point, then exit}
    then
    begin
      GCbound[level,initbranch] := bignum;
      level := level + 1;
      goto 1;
    end
    else                                {Else try next lower bound at previous level}
    begin
      temp := bignum;
      genset := gensize[slevel];
      for a := 1 to genset do
      begin
        sstart[sGC[slevel,a]] := 0;
        sfinish[sGC[slevel,a]] := (sstart[sGC[slevel,a]]+duration[sGC[slevel,a]]);
        sscheduled[sGC[slevel,a]] := FALSE;
        scut[sGC[slevel,a]] := FALSE;
        if sGCbound[slevel,a] < temp
          then
```

```
          begin
            branch := a;
            temp := sGCbound[slevel,a];
          end;
        end;
      end;
  end;
  sGCbound[slevel,branch] := bignum;  {Set lower bound of selected operation = bignum}
  sselect := sGC[slevel,branch];      {Selected operation is to be schedueled next}
  nodecnt := nodecnt + 1;

  if machine[sselect] = 1             {Schedule selected operation on proper machine}
    then sblue[slevel] := sselect
      else sgold[slevel] := sselect;

  newstart(sstart,sfinish,sscheduled,sselect);{Calculate new start times}
  scut[sselect] := FALSE;             {Remove selected operation from cut}

  for i := 1 to n do                  {Select successor operation for next cut}
    if(sscheduled[pred[i,1]] = TRUE)
      and (sscheduled[pred[i,2]] = TRUE)
        and (sscheduled[i] = FALSE)
          then scut[i] := TRUE;

  Generate(slevel,scut,sGC);          {Generate new generating set}

end;
upbound := 0;
for i := 1 to n do                    {Update schedule info with new schedule}
begin
  start[i] := sstart[i];
  finish[i] := sfinish[i];
  blue[i] := sblue[i];
  gold[i] := sgold[i];
  cut[i] := scut[i];
  scheduled[i] := sscheduled[i];
  for j := 1 to m do
  begin
    GC[i,j] := sGC[i,j];
    GCbound[i,j] := sGCbound[i,j];
  end;
  if (sfinish[i] - due[i]) > upbound  {Calculate new upper bound}
    then upbound := (sfinish[i] - due[i]);
end;
GetTime(Hour, Minute, Second, Sec100);{Get solution time}
solvetime := ((((hour*60)+minute)*60)+second+(sec100*0.01));
```

```
termnode := termnode + 1;
level := n+1;
1: end;


{*** REINITIALIZE PROCEDURE -- Resets all variables to inital status ***}
procedure reinitialize;

begin
done := FALSE;
solution := FALSE;
level := 0;
mtard := 0;
nodecnt := 0;
termnode := 0;
upbound := 0;
temp := 0;
lowbound := 0;
tempbound := 0;
branch := 0;
select := 0;
genset := 0;
for i := 0 to 60 do
begin
  cut[i] := FALSE;
  scheduled[i] := FALSE;
  job[i] := 0;
  machine[i] := 0;
  duration[i] := 0;
  due[i] := 0;
  finish[i] := 0;
  start[i] := 0;
  overdue[i] := 0;
  gensize[i] := 0;
  gold[i] := 0;
  blue[i] := 0;
  for k := 1 to 2 do
    pred[i,k] := 0;
    for k := 0 to m do
    begin
      GC[i,k] := 0;
      GCbound[i,k] := 0;
    end;
end;
end;
```

```
{********************** MAIN ROUTINE ********************************}
begin
asign (indata,'indata.txt');              {open ascii input data file}
reset (indata);
assign (outdata, 'outdata.txt');          {open ascii output data file}
rewrite (outdata);
readln(indata,filend);
while filend < 1 do                       {While not end-of-file do . . .}
begin

gettime(hour, minute, second, sec100); {Get computation start time}
starttime := ((((hour*60)+minute)*60)+second+(sec100*0.01));

{*** INPUT DATA ***}
read(indata, plant, rcf, drcf, rep, n);   {Read schedule identification information}
writeln(outdata,'=========================================');
writeln(outdata,plant,'-PLANT, ',rcf,'% RCF, ',drcf,'% DRCF,  REPLICATION:',rep);
writeln(outdata);                         {Write schedule header}
for i := 1 to n do
begin
  read (indata,job[i]);                   {Read the job number for each operation}
  read (indata,machine[i);                {Read the machine number for each operation}
  read (indata,pred[i,1], pred[i,2]);     {Read the predecessors of each operation}
  read (indata,duration[i]);              {Read the duration of each operation}
  read (indata,due[i]);                   {Read the due date of each operation}
end;

{*** SELECT INITIAL CUT ***}
for i := 1 to n do
  if (pred[i,1] = 0)
    and (pred[i,2] = 0)
      then cut[i] := TRUE;                {All operations without predecessors in initial cut}

{*** ASSIGN EARLIEST START AND FINISH TIMES ***}
start[0] := 0;                            {Null operation starts at 0}
finish[0] := 0;                           {Null operation finishes at 0}
scheduled[0] := TRUE;                     {Null operation is scheduled}

for i := 1 to n do
  finish[i] := start[i] + duration[i];    {For each operation finish = start + duration}

{*** FIND UPPER BOUND ***}
generate(level,cut,GC);                   {Generate initial generating set}
while solution = FALSE do                 {Until a solution is found do for each level . . .}
begin
  temp := bignum;
```

```
  for a:= 1 to genset do
  begin                                  {Find the operation with the smallest lower bound}
    lowbound := mbound(GC[level,a],start,finish,due,scheduled);
    GCbound[level,a] := lowbound;
    if lowbound < temp
      then
      begin
        branch := a;
        temp := lowbound;
      end;
  end;
  GCbound[level,branch] := bignum;       {Remove selected operation from consideration}
  select := GC[level,branch];            {Designate selected operation}
  nodecnt := nodecnt + 1;                {Increment node count}

  if machine[select] = 1
    then blue[level] := select
      else gold[level] := select;        {Record selected operation}

  newstart(start,finish,scheduled,select);{Calculate new start times}
  cut[select] := FALSE;                  {Remove selected operation from cut}

  for i := 1 to n do
    if (scheduled[pred[i,1]] = TRUE)
      and (scheduled[pred[i,2]] = TRUE)
        and (scheduled[i] = FALSE)
          then cut[i] := TRUE;           {Determine next cut}

  Generate(level,cut,GC);                {Generate next generating set}
end;

for i := 1 to n do
  if (finish[i] - due[i]) > upbound
    then upbound := (finish[i] - due[i]); {When solution achieved, calculate upper bound}

gettime(hour, minute, second, sec100);  {Get solution time}
solvetime := ((((hour*60)+minute)*60)+second+(sec100*0.01));
termnode := termnode + 1;               {Increment number of terminal nodes}

{*** BACKTRACKING ROUTINE ***}
level := level - 1;                     {Decrement level}
While level > 0 do                      {While level greater than zero do . . .}
begin;
 genset := gensize[level];              {Determine size of generating set at this level}
 tempbound := upbound;
 for i := 1 to genset do
```

```
  scheduled[GC[level,i]] := FALSE;    {Unschedule operations at this level}
 for i := 1 to genset do              {Determine lowest lower bound at this level}
  if (GCbound[level,i] < tempbound)
    then
      begin
        branch := i;
        tempbound := GCbound[level,i];
      end;
 if tempbound < upbound then          {If lower bound <upper bound then goto SOLVE}
  solve(level, start, finish, blue, gold, solution, cut, scheduled, GC, GCbound);
level := level - 1;                   {Decrement level}
end;

{*** OUTPUT ROUTINE ***}
for i := 1 to n do                    {For each operation do . . .}
begin
  overdue[i] := finish[i] - due[i];   {Calculate tardiness}
  if overdue[i] < -240                {If > ½ shipping buffer early}
    then overdue[i] := 0              {Operation not tardy}
    else overdue[i] := overdue[i]+480; {Operation tardy (add buffer)}
end;
mtard := 0;
for i := 1 to n do
    then mtard := overdue[i];
mtard := mtard/480;                   {Translate maximum tardiness to days}
if mtard - trunc(mtard) > 0
  then mtard := trunc(mtard) + 1;     {Round up to nearest whole day}

writeln(outdata,'          Blue Machine Schedule');
writeln(outdata,'operation  job  start  finish  due  tardy');
for i := 1 to n do
  if blue[i] > 0
    then
    begin
write(outdata,blue[i]:5);                    {Output operation number}
write(outdata,job[blue[i]]:9);               {Output associated job number}
write(outdata,start[blue[i]]:9:1);           {Output operation start time}
write(outdata,finish[blue[i]]:8:1);          {Output operation finish time}
write(outdata,due[blue[i]]:8);               {Output operation due date}
writeln(outdata,overdue[blue[i]]:8:1);       {Output operation tardiness}
end;
writeln(outdata);
writeln(outdata,'          Gold Machine Schedule');
writeln(outdata,'operation  job  start  finish  due  tardy');
for i := 1 to n do
  if gold[i] > 0
```

```
  then
begin
write(outdata,gold[i]:5);                    {Output operation number}
write(outdata,job[gold[i]]:9);               {Output associated job number}
write(outdata,start[gold[i]]:9:1);           {Output operation start time}
write(outdata,finish[gold[i]]:8:1);          {Output operation finish time}
write(outdata,due[gold[i]]:8);               {Output operation due date}
writeln(outdata,overdue[gold[i]]:8:1);       {Output operation tardiness}
end;
writeln(outdata);
writeln(outdata,'MAXIMUM TARDINESS = ',mtard:3:3,' days') ;
writeln(outdata);
GetTime(Hour, Minute, Second, Sec100);       {Get finish time}
finishtime := ((((hour*60)+minute)*60)+second+(sec100*0.01))-solvetime;
solvetime := solvetime-starttime;            {Calculate solution time}
checktime := finishtime-solvetime;           {Calculate verification time}
writeln(outdata,'SOLUTION TIME =',solvetime:7:1,'seconds');
writeln(outdata,'VERIFICATION TIME = ',checktime:7:1,' seconds');
writeln(outdata);
writeln(outdata,'     NODES:',nodecnt:7:1,'  TERMINAL NODES:',termnode);
writeln(outdata,'=======================================');
writeln(outdata);
reinitialize;                                {Reinitialize all variables}
readln(indata, filend);                      {Read filend status variable}
end;
end;
```

Appendix C: Comparison of Solutions

Plant Type	%RCF	%ΔRCF	Repetition	Solution Type	Maximum Tardiness (Days)	Disaster Blue Solution (Days)	Disaster Gold Solution (Days)
V	105	0	1	Optimal	5	6	5
V	105	0	2	Optimal	4	4	4
V	105	0	3	Optimal	3	4	3
V	105	0	4	Optimal	8	8	8
V	105	25	1	Optimal	8	9	8
V	105	25	2	Optimal	5	11	6
V	105	25	3	Optimal	5	5	5
V	105	25	4	Optimal	11	11	11
V	105	50	1	Optimal	10	11	10
V	105	50	2	Optimal	7	13	8
V	105	50	3	Optimal	7	7	7
V	105	50	4	Optimal	13	13	13
V	115	0	1	Optimal	6	7	6
V	115	0	2	Optimal	5	6	5
V	115	0	3	Optimal	4	4	4
V	115	0	4	Optimal	9	9	9
V	115	25	1	Optimal	9	10	9
V	115	25	2	Optimal	6	13	7
V	115	25	3	Optimal	6	6	6
V	115	25	4	Optimal	12	12	12
V	115	50	1	Optimal	12	13	12
V	115	50	2	Optimal	9	14	9
V	115	50	3	Optimal	9	9	9
V	115	50	4	Optimal	15	15	15
V	125	0	1	Optimal	7	8	7
V	125	0	2	Optimal	6	6	6
V	125	0	3	Optimal	4	5	5

Plant Type	%RCF	%ΔRCF	Repetition	Solution Type	Maximum Tardiness (Days)	Disaster Blue Solution (Days)	Disaster Gold Solution (Days)
V	125	0	4	Optimal	10	10	10
V	125	25	1	Optimal	10	10	10
V	125	25	2	Optimal	7	13	8
V	125	25	3	Optimal	7	7	7
V	125	25	4	Optimal	13	14	13
V	125	50	1	Optimal	13	14	13
V	125	50	2	Optimal	10	16	10
V	125	50	3	Optimal	10	10	10
V	125	50	4	Optimal	6	16	16
A	105	0	1	Heuristic	4	5	4
A	105	0	2	Heuristic	2	3	3
A	105	0	3	Heuristic	3	3	2
A	105	0	4	Heuristic	2	4	4
A	105	25	1	Optimal	5	6	5*
A	105	25	2	Optimal	5	5	5*
A	105	25	3	Optimal	5	5	5*
A	105	25	4	Optimal	5	8	5*
A	105	50	1	Optimal	7	8	7*
A	105	50	2	Optimal	7	8	7*
A	105	50	3	Optimal	7	8	7*
A	105	50	4	Optimal	7	10	7*
A	115	0	1	Heuristic	4	5	4
A	115	0	2	Optimal	3	3	4
A	115	0	3	Heuristic	3	3	3
A	115	0	4	Heuristic	3	4	4
A	115	25	1	Heuristic	6	7	6*
A	115	25	2	Optimal	6	6	6*
A	115	25	3	Optimal	6	8	6*
A	115	25	4	Optimal	6	9	6*

Plant Type	%RCF	%ΔRCF	Repetition	Solution Type	Maximum Tardiness (Days)	Disaster Blue Solution (Days)	Disaster Gold Solution (Days)
A	115	50	1	Optimal	9	9	9*
A	115	50	2	Optimal	9	9	9*
A	115	50	3	Optimal	9	9	9*
A	115	50	4	Optimal	9	10	9*
A	125	0	1	Heuristic	4	6	4
A	125	0	2	Optimal	4	4	5
A	125	0	3	Heuristic	4	4	4
A	125	0	4	Heuristic	5	5	5
A	125	25	1	Optimal	7	8	7*
A	125	25	2	Optimal	7	8	7*
A	125	25	3	Optimal	7	8	7*
A	125	25	4	Optimal	7	9	7*
A	125	50	1	Optimal	10	12	10*
A	125	50	2	Optimal	10	11	10*
A	125	50	3	Optimal	10	12	10*
A	125	50	4	Optimal	10	12	10*
T	105	0	1	Optimal	4	4	8
T	105	0	2	Optimal	5	8	6
T	105	0	3	Heuristic	3	3	3
T	105	0	4	Optimal	3	6	4
T	105	25	1	Optimal	5	8	6
T	105	25	2	Optimal	8	10	8
T	105	25	3	Optimal	5	12	5*
T	105	25	4	Heuristic	5	10	6
T	105	50	1	Optimal	7	11	7
T	105	50	2	Optimal	10	11	10*
T	105	50	3	Optimal	7	14	7*
T	105	50	4	Heuristic	8	12	7*
T	115	0	1	Optimal	5	5	11

Plant Type	%RCF	%ΔRCF	Repetition	Solution Type	Maximum Tardiness (Days)	Disaster Blue Solution (Days)	Disaster Gold Solution (Days)
T	115	0	2	Optimal	6	7	7
T	115	0	3	Heuristic	4	4	4
T	115	0	4	Heuristic	4	7	5
T	115	25	1	Heuristic	6	9	7
T	115	25	2	Optimal	9	12	9*
T	115	25	3	Optimal	6	13	6*
T	115	25	4	Heuristic	7	11	6*
T	115	50	1	Optimal	9	13	9
T	115	50	2	Optimal	12	13	12*
T	115	50	3	Optimal	9	16	9*
T	115	50	4	Heuristic	9	14	9*
T	125	0	1	Heuristic	5	6	12
T	125	0	2	Optimal	8	10	8
T	125	0	3	Heuristic	5	5	5
T	125	0	4	Heuristic	5	8	6
T	125	25	1	Optimal	7	11	7
T	125	25	2	Optimal	11	12	11
T	125	25	3	Optimal	7	7	7*
T	125	25	4	Heuristic	8	12	7*
T	125	50	1	Optimal	10	14	10
T	125	50	2	Optimal	14	14	13*
T	125	50	3	Optimal	10	17	10*
T	125	50	4	Heuristic	10	15	10*

* Non-interactive constraints

Appendix D: TOC Job Shop Algorithm Statistics

Plant Type	%RCF	%ΔRCF	Repetition	Optimal Solution	Solution Time (Seconds)	Verification Time (Seconds)	Terminal Nodes	Total Nodes
V	105	0	1	Yes	0.3	0.1	2	30
V	105	0	2	Yes	0.2	0.1	1	20
V	105	0	3	Yes	0.2	0.2	1	20
V	105	0	4	Yes	0.2	0.1	1	20
V	105	25	1	Yes	0.7	0.1	5	79
V	105	25	2	Yes	0.4	0.1	3	45
V	105	25	3	Yes	0.9	0.1	6	106
V	105	25	4	Yes	0.2	0.0	1	21
V	105	50	1	Yes	0.2	0.1	1	20
V	105	50	2	Yes	0.2	0.1	2	27
V	105	50	3	Yes	0.1	0.4	3	50
V	105	50	4	Yes	0.2	0.0	1	21
V	115	0	1	Yes	0.3	0.0	2	30
V	115	0	2	Yes	0.2	0.1	1	20
V	115	0	3	Yes	0.2	0.1	1	20
V	115	0	4	Yes	0.2	0.0	1	20
V	115	25	1	Yes	0.5	0.1	5	59
V	115	25	2	Yes	0.3	0.1	3	45
V	115	25	3	Yes	0.6	0.1	5	83
V	115	25	4	Yes	0.2	0.0	1	21
V	115	50	1	Yes	0.2	0.1	1	20
V	115	50	2	Yes	0.3	0.0	2	27
V	115	50	3	Yes	0.3	0.1	2	35
V	115	50	4	Yes	0.2	0.1	1	21
V	125	0	1	Yes	0.2	0.1	2	30
V	125	0	2	Yes	0.2	0.1	1	20
V	125	0	3	Yes	0.2	0.1	1	20
V	125	0	4	Yes	0.2	0.1	1	20
V	125	25	1	Yes	0.4	0.1	3	47

Plant Type	%RCF	%ΔRCF	Repetition	Optimal Solution	Solution Time (Seconds)	Verification Time (Seconds)	Terminal Nodes	Total Nodes
V	125	25	2	Yes	0.3	0.1	3	45
V	125	25	3	Yes	0.6	0.0	4	66
V	125	25	4	Yes	0.2	0.1	1	20
V	125	50	1	Yes	0.2	0.1	1	20
V	125	50	2	Yes	0.2	0.1	1	20
V	125	50	3	Yes	0.3	0.1	2	35
V	125	50	4	Yes	0.2	0.1	1	20
A	105	0	1	No	24.6	275.4	14	5128
A	105	0	2	No	7.7	292.3	4	2114
A	105	0	3	No	22.9	277.1	11	11004
A	105	0	4	No	151.4	148.8	14	6355
A	105	25	1	Yes	48.6	0.2	15	1066
A	105	25	2	Yes	39.5	0.1	15	565
A	105	25	3	Yes	37.7	0.2	13	759
A	105	25	4	Yes	46.6	0.1	16	912
A	105	50	1	Yes	15.5	0.1	5	192
A	105	50	2	Yes	15.4	0.1	5	192
A	105	50	3	Yes	9.0	0.2	3	105
A	105	50	4	Yes	15.6	0.1	6	206
A	115	0	1	No	19.3	280.8	10	4238
A	115	0	2	Yes	7.8	0.1	4	241
A	115	0	3	No	26.5	273.5	9	13175
A	115	0	4	No	84.8	215.2	11	6530
A	115	25	1	No	31.4	268.9	10	18385
A	115	25	2	Yes	32.2	0.1	11	448
A	115	25	3	Yes	30.8	0.2	10	666
A	115	25	4	Yes	34.7	0.1	12	649
A	115	50	1	Yes	5.9	0.1	2	62
A	115	50	2	Yes	5.9	0.2	2	62
A	115	50	3	Yes	5.9	0.1	2	62
A	115	50	4	Yes	6.1	0.1	3	74

Plant Type	%RCF	%ΔRCF	Repetition	Optimal Solution	Solution Time (Seconds)	Verification Time (Seconds)	Terminal Nodes	Total Nodes
A	125	0	1	No	19.2	280.8	-	15780
A	125	0	2	Yes	7.7	0.1	4	241
A	125	0	3	No	26.6	273.4	9	13108
A	125	0	4	No	70.3	229.7	15	6619
A	125	25	1	Yes	25.3	0.1	8	327
A	125	25	2	Yes	25.1	0.1	8	327
A	125	25	3	Yes	18.6	0.1	6	234
A	125	25	4	Yes	25.4	0.1	9	341
A	125	50	1	Yes	6.0	0.2	2	62
A	125	50	2	Yes	5.9	0.2	2	62
A	125	50	3	Yes	5.9	0.2	2	62
A	125	50	4	Yes	5.9	0.1	2	62
T	105	0	1	Yes	1.5	40.1	2	2369
T	105	0	2	Yes	0.7	0.1	3	65
T	105	0	3	No	0.4	299.6	1	44301
T	105	0	4	Yes	122.5	0.1	4	21957
T	105	25	1	Yes	57.3	0.1	9	3087
T	105	25	2	Yes	0.5	0.1	1	28
T	105	25	3	Yes	0.5	0.1	1	28
T	105	25	4	No	0.6	299.4	1	41927
T	105	50	1	Yes	1.4	0.1	4	85
T	105	50	2	Yes	0.6	0.0	1	28
T	105	50	3	Yes	0.5	0.1	1	28
T	105	50	4	No	0.5	299.5	1	41344
T	115	0	1	Yes	27.2	146.3	2	19767
T	115	0	2	Yes	0.7	0.1	3	65
T	115	0	3	No	0.4	299.6	1	44315
T	115	0	4	No	172.1	127.9	5	40641
T	115	25	1	No	2.9	297.1	8	24066
T	115	25	2	Yes	0.6	0.1	1	28
T	115	25	3	Yes	0.5	0.1	1	29

Plant Type	%RCF	%ΔRCF	Repetition	Optimal Solution	Solution Time (Seconds)	Verification Time (Seconds)	Terminal Nodes	Total Nodes
T	115	25	4	No	0.5	299.6	1	40452
T	115	50	1	Yes	1.1	0.2	3	80
T	115	50	2	Yes	0.5	0.0	1	28
T	115	50	3	Yes	0.6	0.1	1	28
T	115	50	4	No	0.4	299.6	1	40860
T	125	0	1	No	52.6	247.4	3	36251
T	125	0	2	Yes	0.7	0.1	3	65
T	125	0	3	No	0.4	299.6	1	44106
T	125	0	4	No	173.2	126.8	5	40455
T	125	25	1	Yes	2.1	0.1	7	142
T	125	25	2	Yes	0.4	0.1	1	28
T	125	25	3	Yes	0.5	0.1	1	28
T	125	25	4	No	0.5	299.5	1	40516
T	125	50	1	Yes	1.1	0.1	3	67
T	125	50	2	Yes	0.5	0.1	1	28
T	125	50	3	Yes	0.4	0.1	1	28
T	125	50	4	No	0.4	299.6	1	40145

Bibliography

Baker, Kenneth R. *Introduction to Sequencing and Scheduling.* New York: John Wiley and Co, 1974.

-----. "Sequencing Rules and Due-Date Assignments in a Job Shop," *Management Science, 30*: 1093-1104 (September 1984).

Cody, Ronald P. and Jeffrey K. Smith. *Applied Statistics and the SAS Programming Language.* Amsterdam, Netherlands: Elsevrier Science Publishing, 1991.

Conway, Richard W., William L. Maxwell, and Louis W. Miller. *Theory of Scheduling.* Reading MA: Addison-Wesley Publishing Company, 1967.

Cummins, Christopher S., "Mobilization as a Constraint on Operations: Applying the Theory of Constraints," *Air Force Journal of Logistics*: 27-30 (Winter-Spring 1993).

Davis, Edward W. and George E. Heidorn, "An Algorithm for Optimal Project Scheduling Under Multiple Resource Constraints," *Management Science, 17*: B803-B816 (August 1971).

Demmy, W. Steven and Arthur B. Petrini. "The Theory of Constraints: A New Weapon for Depot Maintenance Planning and Control," *Air Force Journal of Logistics, 16*: 6-11 (Summer 1992).

Devore, Jay L. *Probability and Statistics for Engineering and the Sciences*, Pacific Grove CA: Brooks/Cole Publishing Co, 1991.

Fawcett, Stanley E. and John N. Pearson. "Understanding and Applying Constraint Management in Today's Manufacturing Environments," *Production and Inventory Management Journal, 32*: 46-55 (Third Quarter 1991).

Florian, M. P. Trepant, and G. McMahon. "An Implicit Enumeration Algorithm for the Machine Sequencing Problem," *Management Science, 17*: B782-B792 (August 1971).

Fruend, Rudolf J. and Ramon C. Littell. *SAS for Linear Models: A Guide to the ANOVA and GLM Procedures.* Gary NC: SAS Institute Inc, 1981.

Gargeya, Vidyarana B. "An Evaluation of Resource Constraint Measures in a Dual Constrained Job Environment," *Proceedings of the 1992 Annual Meeting of the Decision Sciences Institute*. 1538-1540.

Goldratt, Eliyahu M. and Jeff Cox. *The Goal, A Process of Ongoing Improvement*. Croton-on-Hudson NY: North River Press, 1992.

----- and Robert E. Fox. *The Race*. Croton-on-Hudson NY: North River Press, 1986.

-----. "Computerized Shop Floor Scheduling," *International Journal of Production Research, 26*, No 3: 443-455 (March 1988).

-----. *The Haystack Syndrome*. Croton-on-Hudson NY: North River Press, 1990.

Graves, Stephen C. "A Review of Production Scheduling," *Operations Research Journal, 29*, No 4: 646-667 (July-August 1981).

Heizer, Jay, Barry Render and Ralph M. Stair, Jr. *Production and Operations Methods*. Boston: Allyn and Bacon. 1992.

James, Stewart W. and Bruno A. Mediate Jr. *Benchmark Production Scheduling Problems for Job Shops with Interactive Constraints*. MS thesis, AFIT/GSM/LAS/93S-9. School of Logistics and Acquisition Management, Air Force Institute of Technology (AU), Wright-Patterson AFB OH, September 1993.

Kanet, John J. "MRP 96: Time to Rethink Manufacturing Logistics," *Production and Inventory Management Journal, 29*, No 2: 57-61 (Second Quarter 1988).

Kirk, Roger E. *Experimental Design: Procedures for the Behavioral Sciences*, Monterrey CA: Brook/Cole Publishing Co., 1982

Levin, Richard I. and Charles A. Kirkpatrick. *Quantitative Approaches to Management*. New York: McGraw-Hill Inc, 1975.

Manne, Alan S. "On the Job-Shop Scheduling Problem," *Operations Research, 8*: 219-223 (March 1960).

Neter, John, William Wasserman, and Michael Kutner. *Applied Linear Statistics Models*. Boston: Irwin Publishing, 1990.

Newbold, Robert C. Advisor, Avraham Y. Goldratt Institute, New Haven CT. Attachment on Drum Logic. 27 August 1990.

-----. Personal correspondence. 23 September 1992.

Patterson, James H. "A Comparison of Exact Approaches for Solving the Multiple Constrained Resource Project Scheduling Problem," *Management Science, 30*: 854-867 (July 1984).

Ragatz, Gary L. and Vincent A. Mabert. "A Simulation Analysis of Due Date Assignment Rules," *Journal of Operations Management, 5*: 27-39 (November 1984).

Reynolds, Daniel E., Professor, Air Force Institute of Technology, Wright-Patterson Air Force Base OH. Personal interview, 10 August 1993.

Rose, Coral. Theory of Constraints Instructor for the Avraham Y. Goldratt Institute, New Haven CT. Personal conversation with Major J.V. Simons, Jr., AFIT/LAL. 22 February 1993.

Severs, Jefferson L. *The Disaster Scheduling System: A Review and Case Analysis*. MS thesis, AFIT/GLM/LSM/91S-56. School of Systems and Logistics, Air Force Institute of Technology (AU), Wright-Patterson AFB OH, September 1991 (AD-246758).

Simons, Jacob V. and Richard I. Moore. "The Theory of Constraints Approach to Focused Improvement," *Air Force Journal of Logistics, 16*: 1-5 (Summer 1992).

-----., Associate Professor, Air Force Institute of Technology, Wright-Patterson Air Force Base OH. Personal correspondence. 1 September 1992.

Simpson, Wendell P. III. *A Parallel Exact Solution Procedure for the Resource-Constrainted Project Scheduling Problem*. PhD Dissertation. School of Business, Indiana University, March, 1991.

Stinson, Joel P., Edward W. Davis, and Basheer M. Khumawala, "Multiple Resource--Constrained Scheduling Using Branch and Bound," *AIIE Transactions*, 10: 252-259 (September 1978).

Swartz, Steven., Metrics Program Manager, Production Policy Division, Directorate of Logistics, Headquarters Air Force Materiel Command, Wright-Patterson AFB OH. Personal interview. 9 August 1993.

Wight, Oliver W. *MRP II: Unlocking America's Productivity Potential*, Boston: CBI Publishing Co., 1981

Vita

Captain Barak J. Carlson was born on 24 May 1966 in Pueblo Colorado. He graduated from Upper St Clair High School in Upper St Clair Pennsylvania in 1984. He attended the University of Colorado and graduated with a Bachelor of Science Degree in Aerospace Engineering in May 1988. He received a reserve commission in the USAF through the Reserve Officer Training Corps. He began his career as a Flight Test Manager at the F-16 System Program Office at Wright-Patterson AFB OH. He transitioned to Program Management in the Close Air Support F-16 program in 1991. He entered the School of Logistics and Acquisition Management, Air Force Institute of Technology, in May of 1992.

Permanent Address: 3295 Nan Pablo Dr
Melbourne FL 32325

Captain Christopher A. Lettiere was born on 10 April 1966 in Hartford, Connecticut. He graduated from Conard High School in West Hartford Connecticut in 1984. He attended the University of Connecticut and graduated with a Bachelor of Science Degree in Electrical Engineering in December 1988. He received a regular commission in the USAF through the Reserve Officer Training Corps and reported for active duty at Holloman AFB, New Mexico in April 1989. He began as a Flight Test Engineer for the 6585th Test Group where he was responsible for airborne testing of Global Positioning System User Equipment and Fiber Optic Gyro Inertial Navigation Systems. In 1991 he was selected for the GPS Integration Management team where he supported the F-15E, F-16C/D Block 50, and B-2 GPS integration efforts. He entered the School of Logistics and Acquisition Management, Air Force Institute of Technology, in May of 1992.

Permanent Address: 5 Kimberly Rd
West Hartford CT 06107

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington Headquarters Services, Directorate for information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188), Washington, DC 20503

1. AGENCY USE ONLY *(Leave blank)*	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	September 1993	Master's Thesis

4. TITLE AND SUBTITLE
A COMPARISON OF THE DISASTER™ SCHEDULING SOFTWARE WITH A SIMULTANEOUS SCHEDULING ALGORITHM FOR MINIMIZING MAXIMUM TARDINESS IN JOB SHOPS

5. FUNDING NUMBERS

6. AUTHOR(S)
Captain Barak J. Carlson
Captain Christopher A. Lettiere

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Air Force Institute of Technology, WPAFB OH 45433-6583

8. PERFORMING ORGANIZATION REPORT NUMBER
AFIT/GSM/LAS/93S-3

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)
HQ AFMC/LG-PP, WPAFB OH

10. SPONSORING / MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES

12a. DISTRIBUTION / AVAILABILITY STATEMENT
APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED

12b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 words)*

The Theory of Constraints (TOC) is the foundation for a computerized scheduling system called DISASTER™. Although this system has proven successful in many manufacturing settings, it has potential limitations due to the sequential heuristic process by which it schedules constraints. The objective of this thesis was to determine the extent to which these limitations impact the due date performance of schedules created by DISASTER™. This objective was addressed by developing an algorithm to simultaneously schedule multiple constraints in a job shop environment and provide the optimal schedule for minimized maximum tardiness. This algorithm was used to obtain solutions for a matrix of job shop problems, which were compared with solutions obtained by using DISASTER™. This comparison showed that DISASTER™ is capable of producing nearly optimal solutions for minimized maximum tardiness, but that this capability is highly dependent on proper constraint sequencing.

14. SUBJECT TERMS
Theory of Constraints, job shop scheduling, production, scheduling, computer programs, tardiness, branch and bound

15. NUMBER OF PAGES
105

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
Unclassified	Unclassified	Unclassified	UL

AFIT Control Number AFIT/GSM/LAS/93S-3

AFIT RESEARCH ASSESSMENT

The purpose of this questionnaire is to determine the potential for current and future applications of AFIT thesis research. Please return completed questionnaires to: DEPARTMENT OF THE AIR FORCE, AIR FORCE INSTITUTE OF TECHNOLOGY/LAC, 2950 P STREET, WRIGHT PATTERSON AFB OH 45433-7765

1. Did this research contribute to a current research project?

a. Yes b. No

2. Do you believe this research topic is significant enough that it would have been researched (or contracted) by your organization or another agency if AFIT had not researched it?

a. Yes b. No

3. The benefits of AFIT research can often be expressed by the equivalent value that your agency received by virtue of AFIT performing the research. Please estimate what this research would have cost in terms of manpower and/or dollars if it had been accomplished under contract or if it had been done in-house.

Man Years _____ $ _____

4. Often it is not possible to attach equivalent dollar values to research, although the results of the research may, in fact, be important. Whether or not you were able to establish an equivalent value for this research (3, above) what is your estimate of its significance?

a. Highly Significant b. Significant c. Slightly Significant d. Of No Significance

5. Comments

_____ _____

Name and Grade Organization

_____ _____

Position or Title Address

DEPARTMENT OF THE AIR FORCE
AFIT/LAC Bldg 641
2950 P St
45433-7765

OFFICIAL BUSINESS

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY MAIL

FIRST CLASS MAIL PERMIT NO. 1006 DAYTON OH

POSTAGE WILL BE PAID BY U.S. ADDRESSEE

Wright-Patterson Air Force Base

AFIT/LAC Bldg 641
2950 P St
Wright-Patterson AFB OH 45433-9905